

Boost Converter Supervision Interface



Contacts :

- **Citrocaen Embedded Systems division** – référent technique systèmes embarqués et intégration – coordinateur projet "boost converter supervision interface" : Descoubes Hugo : hugo.descoubes@ensicaen.fr
- **Citrocaen Signal division** – référent technique traitement du signal : Miloud Frikel - mfrikel@ensicaen.fr
- **Citrocaen Automatic division** – référent technique automatique et système : Gehan Olivier : gehan@greyc.ensicaen.fr
- **Citrocaen Network division** – référent technique réseaux et télécoms : Lefebvre Philippe : philippe.lefebvre@ensicaen.fr

Coordonnées :

CITROCAEN – ENSICAEN corp.
6 Bd Maréchal Juin
14050 Caen cedex
+033 02 31 27 61

CONFIDENTIAL

PLAN DOCUMENT

Document présentant le cahier des charges, spécifications, contraintes technologiques et échéances à respecter dans le cadre du nouveau et ambitieux projet lancé CITROCAEN et soutenu par le consortium "for a better world" (RENAUD inc., PORCHE corp., MERCIDES BEN corp.). Ce projet s'insère donc dans un plus vaste programme de rénovation des solutions de supervision des systèmes de puissances pour nos systèmes autonomes et systèmes automobiles.

1. PRESENTATION GENERALE

- 1.1. Contexte
- 1.2. Partenariats et contraintes technologiques

2. MODULE DE PUISSANCE

- 2.1. Électronique de puissance
- 2.2. Système de mesure
- 2.3. Automatique de commande

3. SYSTEME DE COMMANDE TEMPS REEL

- 3.1. Système temps réel
- 3.2. Interfaces

4. BUS CAN

- 4.1. Architecture réseau
- 4.2. Protocole plug-and-play

5. SYSTEME DE SUPERVISION

- 5.1. Processeur Application
- 5.2. Interfaces

6. INTERFACE UTILISATEUR

- 6.1. Système de contrôle/supervision
- 6.2. Interface ordinateur
- 6.3. Interface Android

7. DOCUMENTATIONS TECHNIQUES

- 7.1. Schémas carte puissance
- 7.2. liens utiles

CONFIDENTIAL

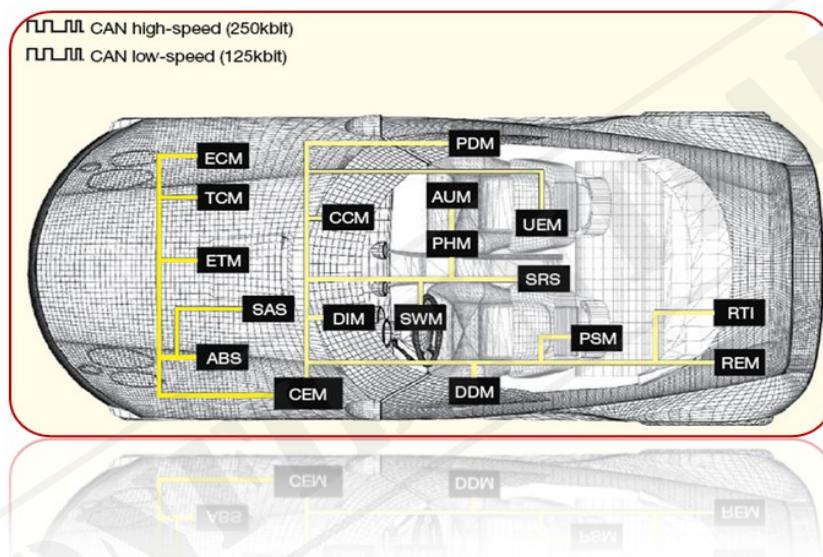
1. PRESENTATION GENERALE

1.1. Contexte

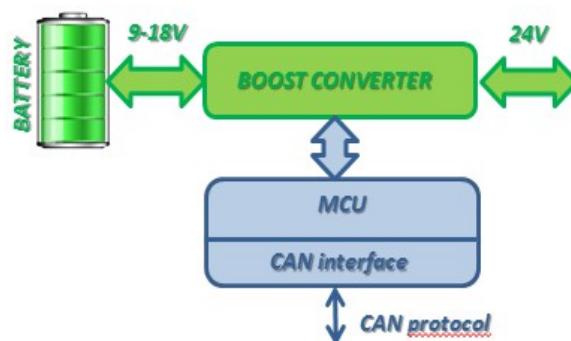
Nous constatons depuis plusieurs années de grands problèmes de stabilité et de durabilité des systèmes de distribution et de conversion de puissance dans nos solutions électroniques embarquées. En effet, la durée de vie de tels systèmes est fortement assujettie aux technologies déployées et forts courant amenés à traverser ces systèmes de puissance.

Dans ce contexte, pour un usage interne aux équipes techniques de développement de CITROCAEN, **nous souhaitons mettre en œuvre un système, non intrusif, de supervision de nos modules de puissance depuis un ordinateur ou une tablette Android**. La solution proposée devra donc s'adapter aux contraintes technologiques des systèmes existants et solutions techniques déjà déployées.

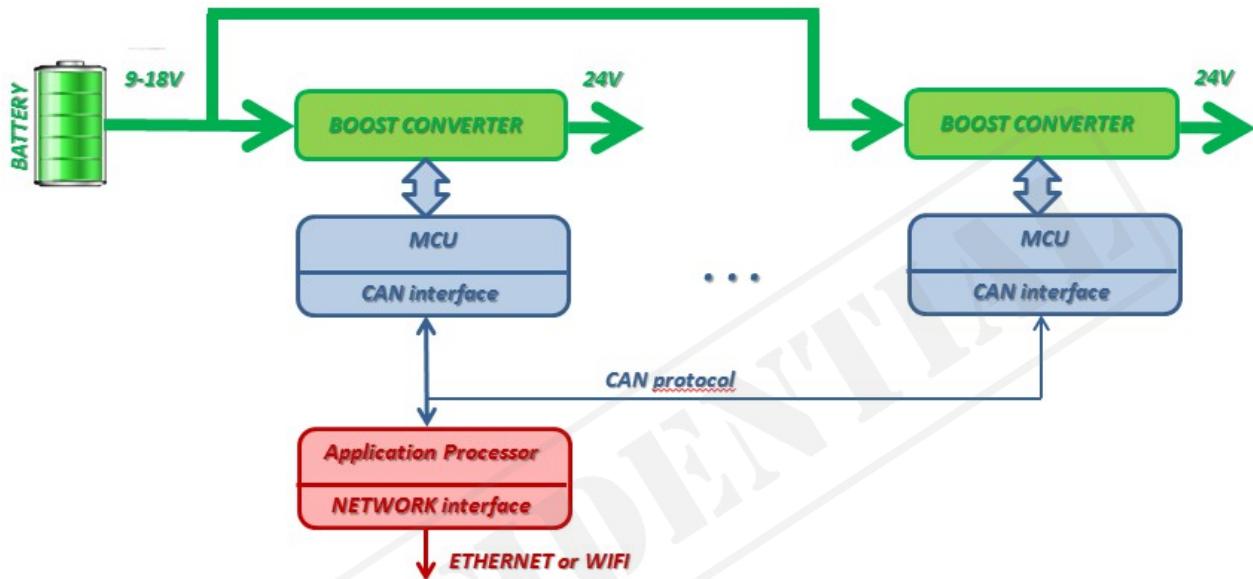
Rappelons l'architecture typique d'un système électronique automobile :



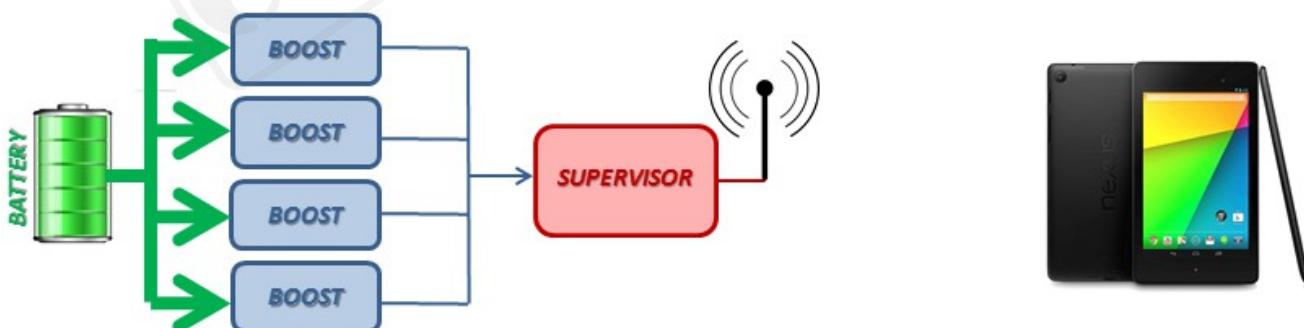
Chacun des modules externalisés (ECM, TCM, ABS...) sur le véhicule est interfacé par un **réseau industriel CAN** et possède son propre régulateur de tension. Pour certains de ces éléments, non avons besoin d'une élévation locale du niveau de tension (convertisseur élévateur ou boost converter). Nous souhaitons donc mettre en œuvre une supervision voir contrôle de ces régulateurs de puissance pour une meilleure maintenabilité et évolutivité (évolution de futurs standards) :



Chaque entité (*boost converter + MCU/CAN interface*) doit être vu comme une entité autonome. Un processeur application de contrôle/supervision devra être **capable d'interfacer jusqu'à quatre modules (*boost converter + MCU/CAN interface*)**, sachant que le firmware devra être pensé pour être facilement évolutif :



Le processeur application devra quant à lui proposer une interface réseau classique, ethernet vers un ordinateur (non prioritaire pour le projet concernant la partie IHM) et **WIFI en mode Ad-Hoc (infrastructure si possible)** afin de pouvoir contrôler et superviser les différents procédés à distance depuis une **tablette Android**. Observons l'architecture simplifiée de l'application :



Voici le cahier des charges simplifié du projet demandé (détailé par la suite) :

- Développer un algorithme de commande avancé pour piloter le module de puissance **BOOST CONVERTER** (convertisseur fourni)
- Intégrer l'algorithme de commande sur **MCU** en cohabitation avec système d'exploitation temps réel
- Établir une interface **CAN** de communication avec sur-couche protocolaire **plug-and-play**
- Développer le firmware de supervision sur processeur application
- Établir une interface de **WIFI Ad-Hoc ou infrastructure de communication**
- Développer une application **Android** comme démonstrateur

1.1. Partenariats et contraintes technologiques

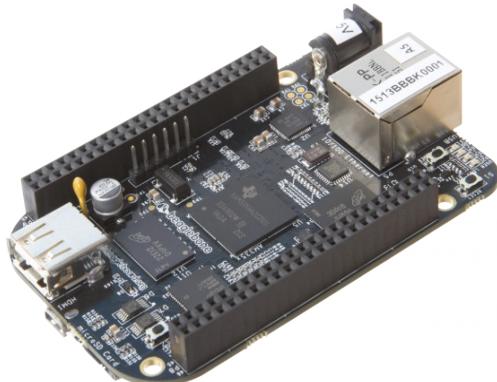
La solution proposée devra donc s'intégrer aux systèmes existants. Pour des raisons de confidentialité, les firmwares déjà existants et actuellement en service ne pourront pas vous être fournis, néanmoins nous souhaitons continuer nos développements sur les mêmes plateformes matérielles et logicielles. Nous travaillons maintenant depuis plusieurs années sur solutions à bas coût Microchip (PIC18, PIC24, dsPIC et PIC32) et processeurs application SoC de Texas Instruments (OMAP, AM335x) pour une meilleure polyvalence et flexibilité.

Les solutions logicielles déployées font appel au monde de l'OpenSource compte tenu des volumes en jeu (royalties free). Nous travaillons depuis près de quatre années sur **RTOS (système d'exploitation temps réel) FreeRTOS** et depuis près de 9 ans sur **solutions Linux Embarqué**. Nous vous imposons les plateformes matérielles et exécutif portés.

MCU avec firmware temps réel (PIC32MX795F512L et FreeRTOS). Entre deux et quatre plateformes de développement vous seront fournies. Il s'agit de **kit de développement bas coût proposés par Olimex (PIC32 MAXI WEB)** :



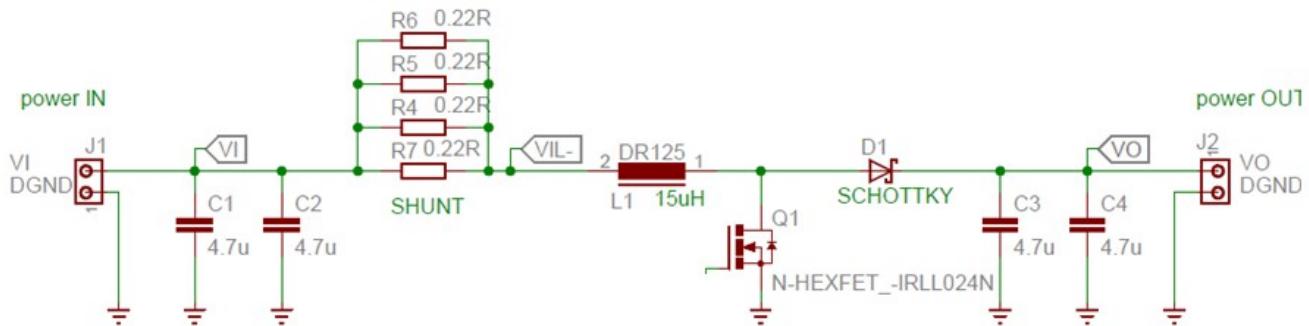
Concernant le processeur application de supervision, deux solutions s'offrent à vous. Utiliser la même plateforme que le processeur temps réel (PIC32MX795F512L et FreeRTOS) ou un **SoC de TI (AM335x) sous Linux Embarqué**. Une société de sous-traitance est actuellement en cours de développement d'une distribution custom adaptée à nos besoins (CAN interface support, Acces Point support, AdHoc support ...). Néanmoins, rien ne garanti l'arrivée de cette solution Linux custom dans les délais (retard actuel de plusieurs semaines). Plateforme Sitara AM335x de TI (**beaglebone black**) :



2. MODULE DE PUISSANCE

2.1. Électronique de puissance

La tâche première du projet est avant tout la commande numérique temps réel d'un **convertisseur élévateur de tension** (boost converter). Schéma électrique du convertisseur à piloter :



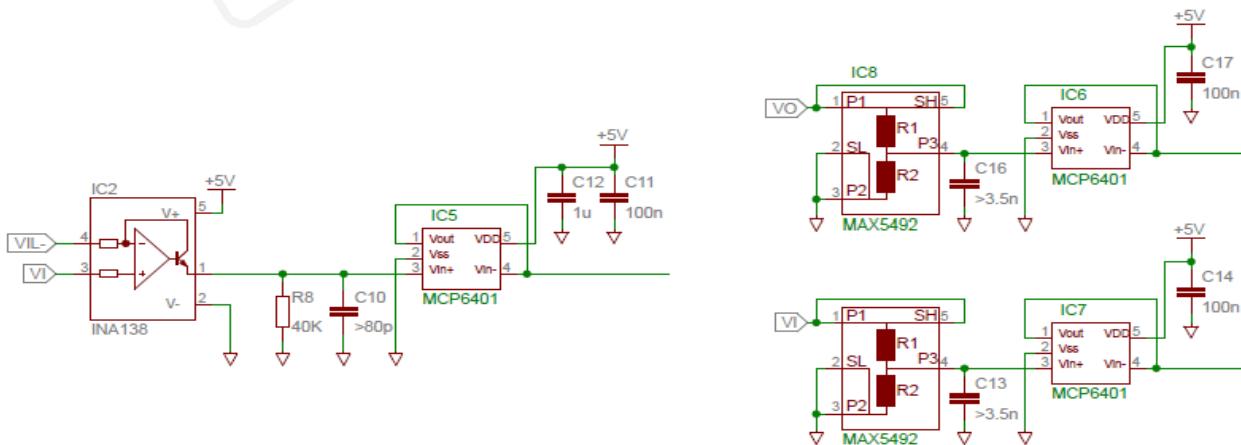
Observons les spécifications électriques du montage et du cahier des charges. La structures présentées ci-dessus fonctionne sur les gammes suivantes :

- **Puissance en sortie : 14,4W**
- **Tension d'entrée (sur batterie) : 9-18V (3A max), ~12V typique**
- **Tension de sortie : 24V (ou 36V max – 400mA)**
- **Fréquence de découpage : 500KHz**

2.2. Système de mesure

Compte tenu des gammes de courants et tensions en jeu, les mesures ne proposent aucune solution d'isolation galvanique (isolation électrique puissance/signal). Trois mesures distinctes sont réalisées sur le montage (attention à la chaîne de conditionnement – pont diviseurs MAX5492 – 10KOhm – 1:1 vers 10:1) :

- **Tension d'entrée : VI**
- **Courant d'entrée : IL (mesure différentielle sur shunt)**
- **Tension de sortie : VO**



2.3. Automatique de commande

Nous vous demandons le développement d'une **loi numérique de commande** (avancée). Attention, les systèmes boost sont des **systèmes fortement non-linéaire et oscillatoires**. Nous demandons dans cette étude suivie d'une intégration sur MCU, une très grande rigueur quant à la **modélisation du procédé, la technique d'identification, la synthèse de la loi de commande et l'intégration**. De plus, le système à commander possède une **dynamique extrêmement rapide** et imposera donc une grande rigueur durant la phase d'intégration (temps réel dur). Sans la mise en œuvre de l'interface CAN (vers processeur de supervision), le système devra être entièrement autonome et assurer l'asservissement temps réel.

Cahier des charges :

- **Les échéances seront fixées en cours de projet**
- **Modélisation Mathématique du procédé (modèle aux valeurs moyennes) sur modèle réel avec résistance série de l'inductance (à envoyer en cours de projet).**
- **Procédures d'identification (à envoyer en cours de projet).**
- **Modélisation MATLAB/SIMULINK du procédé (à envoyer en cours de projet).**
- **Synthèse sous MATLAB/SIMULINK d'une loi de commande avancée (à envoyer en cours de projet).**
- **Intégration C de l'algorithme (à envoyer en cours de projet).**

Spécifications de l'algorithme de commande (dynamique, poursuite, régulation ...) :

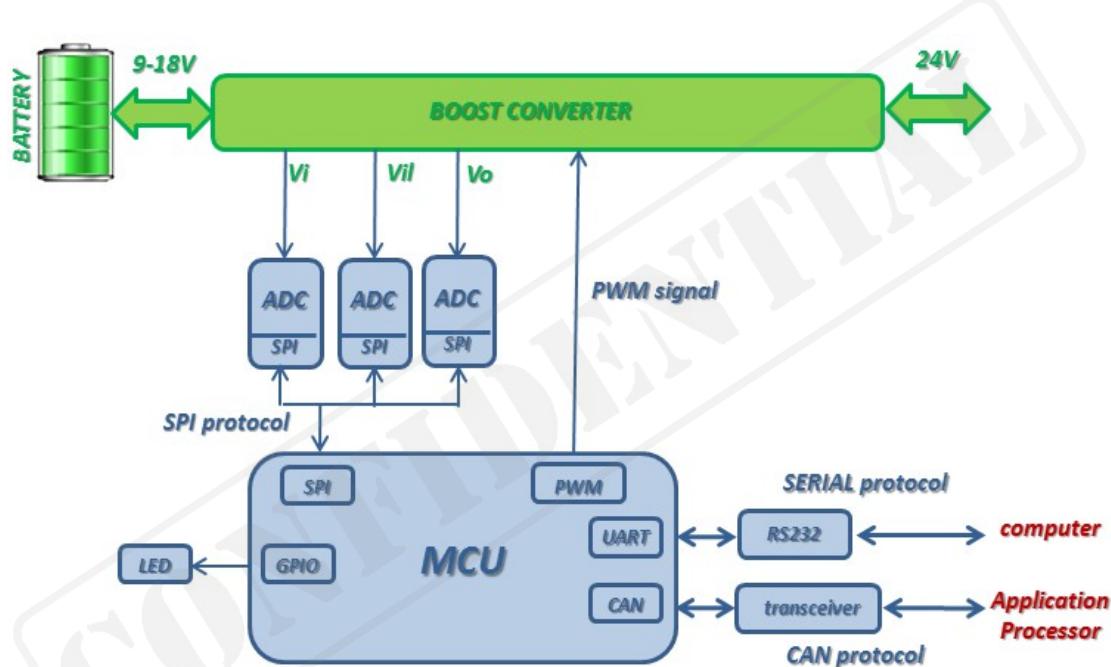
- **Les échéances seront fixées en cours de projet**
- **Les spécifications (poursuite, régulation ...) vous seront fournies le plus rapidement possible (début de projet).**

3. SYSTEME DE COMMANDE TEMPS REEL

3.1. Système temps réel

Le MCU utilisé devra être un PIC32MX795F512L proposé par Microchip et devra porter FreeRTOS (adaptation facile à nos solutions déjà existantes). Celui-ci devra assumer de très forte contraintes temps réel (partie automatique commande) ainsi que plusieurs interfaces généralistes et de communication. Observons les :

- **Interfaces SPI** : 3 composants à piloter
- **Interface PWM** : contrôle driver MOSFET
- **Interface GPIO** : LED utilisateur
- **Interface UART** : interface RS232 vers PC (développement interface shell)
- **Interface CAN** : vers processeur application (développement protocole plug-and-play)

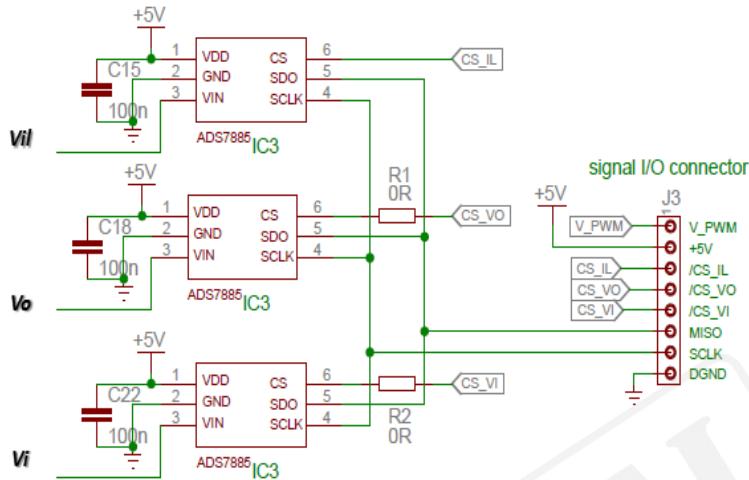


3.2. Interfaces

L'interface CAN sera présentée plus en détail par la suite, notamment la sur-couche protocolaire plug-and-play qui vous est demandé de développer. Intéressons nous aux autres interfaces :

- **Interface PWM** : fréquence de découpage 500KHz (lié au design de la structure de puissance). Utilise le driver de MOSFET MCP1402 de Microchip.
- **Interface LED** : L'application devra détecter la présence ou non d'une charge connectée. En l'absence de charge utile (courant de sortie 0A) la LED devra clignoter avec une périodicité de 500ms (500ms sec allumée, 500ms sec éteinte ...). En cas de présence d'une charge, la LED restera allumée.

- **Interface SPI** : pilotage de trois ADC SPI ADS7884 3Msps, 1canal de Analog Device. Les trois ADC assurent les conversions analogiques/numériques de V_{il} (image de i_l , courant d'entrée), V_i (tension d'entrée) et V_o (tension de sortie) . Attention au conditionnement du signal.



- **Interface UART** : L'application devra proposer un shell afin de communiquer voir reconfigurer sans recompilation des variables d'environnement du firmware embarqué. L'application devra proposer une interface de communication (cf. shell) implémentant un jeu de commande prédéfini dans le firmware. Jeu de commande à supporter :
 - **auto** : passe en mode automatique avec algorithme de commande. L'algorithme de commande est activé et doit garantir les spécifications du cahier des charges de la partie automatique de commande. La commande **pwm** est alors *inactive* . La commande **ref** permet de modifier la valeur de la consigne de tension.
 - **manual** : passe en mode manuel sans algorithme de commande. Seul le signal **PWM** est généré avec un rapport cyclique de 0% par défaut. Modification du rapport cyclique via la commande **pwm**. La commande **ref** est alors *inactive*.
 - **ref** : modification de la consigne de tension. La valeur entière ne peut être que : 18V, 24V, 30V ou 36V. Retourne un message d'erreur en cas de valeur erronée.
 - **pwm** : modification de la valeur du rapport cyclique du signal **PWM**. Valeur entière comprise entre 0 et 100. Retourne un message d'erreur en cas de valeur erronée.
 - **meas** : retourne les 3 dernières valeurs des grandeurs analogiques converties (i_l en A, V_o en V et V_i en V)
 - **can** : affiche la configuration de la communication CAN (débits données, taille charge utile, filtre et masque de filtres).
 - **filter** : modification du filtre associé au module CAN. Entrer 3 caractères hexadécimal. Retourne un message d'erreur en cas de valeur erronée.
 - **help** : retourne le jeu de commandes supportées ainsi qu'une courte aide

L'interface proposée par le MCU devra absolument respecter l'affichage suivant (exemple de d'implémentation du jeu de commandes) :

cf. page suivante

*citroen corp. with your company partnership
boost converter module interface. Enter help to know the supported commands
boost:~# help*

```
auto          start automatic mode and stop manual mode
manual        update voltage reference value with ref command
              start manual mode and stop automatic mode
              update duty cycle value with pwm command
ref           update output voltage reference value
pwm           update duty cycle value
meas          print currents analog values
can           print can controller configuration
filter        update can filter
help          print supported commands
boost:~# auto
automatic mode is running
change voltage reference value with ref command
boost:~# ref
enter desired ouput voltage value : 18 or 24 or 30 or 36 (V)
24
ouput voltage value is updated
boost:~# ref
enter desired ouput voltage value : 18 or 24 or 30 or 36 (V)
23
error: failed value
boost:~#
```

```
boost:~# manual
pwm mode is running
change duty cycle value with pwm command
boost:~# pwm
enter duty cycle value between 0 and 100 :
56
the duty cycle value is updated
boost:~# pwm
enter duty cycle value between 0 and 100 :
113
error: failed value
boost:~# can
can controller configuration :
bus speed 250000 bps
data length 8 bytes
filter 0x002
filter mask 0xFFFF
boost:~# filter
enter can filter value in hexadecimal (3 caracters) :
00A
can filter value is updated
boost:~# filter
enter can filter value in hexadecimal (3 caracters) :
0G01
error: failed value
boost:~# azerty
error: enter help to know the supported commands
boost:~#
```

Sans la mise en œuvre de l'interface CAN (vers processeur de supervision), le système devra être entièrement autonome et assurer l'asservissement temps réel.

Cahier des charges :

- **Les échéances seront fixées en cours de projet**
- **Proposer une modélisation de l'architecture logicielle de l'application (environnement multi-tâches, outils de synchronisation, communication, protection). La modélisation pourra être fournie avec une convention graphique propre à votre entreprise ou sinon devra respecter les conventions du standard SysML (Systems Modeling Languages). A envoyer en cours de projet.**
- **Votre solution logicielle devra être disponible et téléchargeable via internet à chaque moment du projet (mise en place dès le 3 jours de projet). Nous vous demandons d'utiliser des solutions (temporairement) ouverte via dépôt distant sur GitHub (interface familière du côté de nos équipes). Il faudra donc un dépôt GitHub unique pour le firmware propre à cette partie de l'application.**

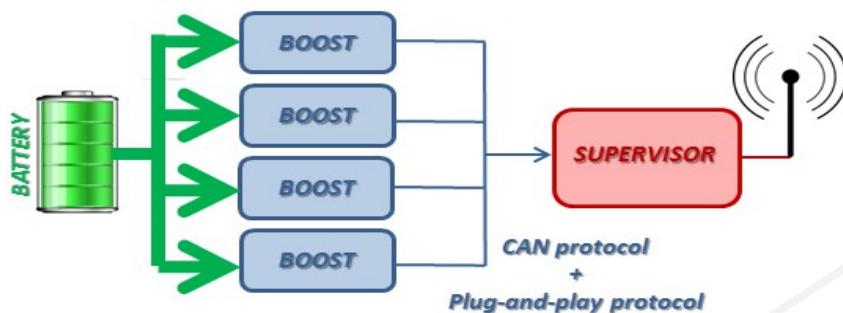
Spécifications :

- **Les parties "acquisition, algorithme et génération du signal de commande" (temps réel dur) ne devront pas surcharger le CPU à hauteur de plus de 60-65%**
- **libération à hauteur de 35-40% de la charge CPU pour le reste de l'application (communications : liaison série, can, led)**

4. BUS CAN

4.1. Architecture réseau

Pour sa grande robustesse justifiant sa très forte utilisation en milieu automobile et industriel, nous implémenterons une solution CAN compatible avec la version CANv2 des spécifications BOSCH. Rappelons l'architecture du réseau lié à notre application.



4.2. Protocole plug-and-play

A l'image du protocole USB, il vous est demandé d'implémenter une sur-couche protocolaire au CAN implémentant une solution plug-and-play. En effet, les modules se veulent autonomes par défaut (si aucune interface CAN, reconfiguration seulement possible par liaison série). Par défaut, le processeur de supervision n'a aucun module BOOST connecté à son bus. Si un module BOOST se connecte, le superviseur doit être capable d'identifier le module (4 au maximum) puis d'établir un protocole d'échange et de contrôle avec le module BOOST (boost converter+MCU). **Prévoir un protocole évolutif pouvant s'adapter dans le futur à plus de 4 modules.**

Cahier des charges :

- **Les échéances seront fixées en cours de projet**
- **Proposer les spécifications de cette sur-couche protocolaire (à envoyer en cours de projet).**
- **Pour des soucis de compatibilité et de criticité, nous travaillons sur ce projet avec 2 entreprises partenaires (dont vous). Vos 2 solutions techniques devront être compatibles (2 sociétés distinctes). Vous devrez entrer en contact avec cette autre société et vous entendre sur les spécifications du protocole. Un ingénieur technique de nos équipes CITROCAEN supervisera les échanges, servira de médiateur et pourra trancher en cas de discorde sur une solution technique. Nous vous mettrons en contact avec les différentes intervenants. Communication par (conf. Call, visio conf ...)**

Spécifications :

- **Protocole évolutif mais utilisant la bande passante du bus de façon optimale**

5. SYSTEM DE SUPERVISION

5.1. Processeur Application

Pour cette partie de l'application, 2 solutions techniques s'offrent à vous en fonction du processeur sur lequel vous travaillez.

- **Solution n°1 :**

Travailler sur le même processeur que celui utilisé pour la partie asservissement du module convertisseur boost. Cette solution est moins intéressante de notre côté car plus difficilement évolutive (processeur moins polyvalent). Néanmoins, à cause d'un délai non respecté d'une autre société de prestation, nous ne pouvons pas vous fournir une solution clé en main à ce stade là du projet (distribution custom).

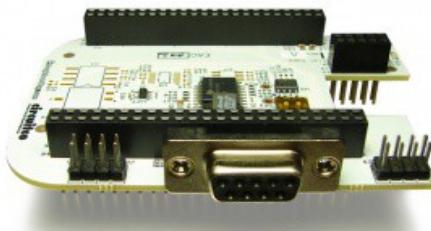
Si le développement est fait sur PIC32MX795F512L, il vous sera également fourni un module d'interface wifi pour le développement de la communication avec la tablette ANDROID. Module wifi proposé par Olimex (MOD WIFI) :



- **Solution n°2 :**

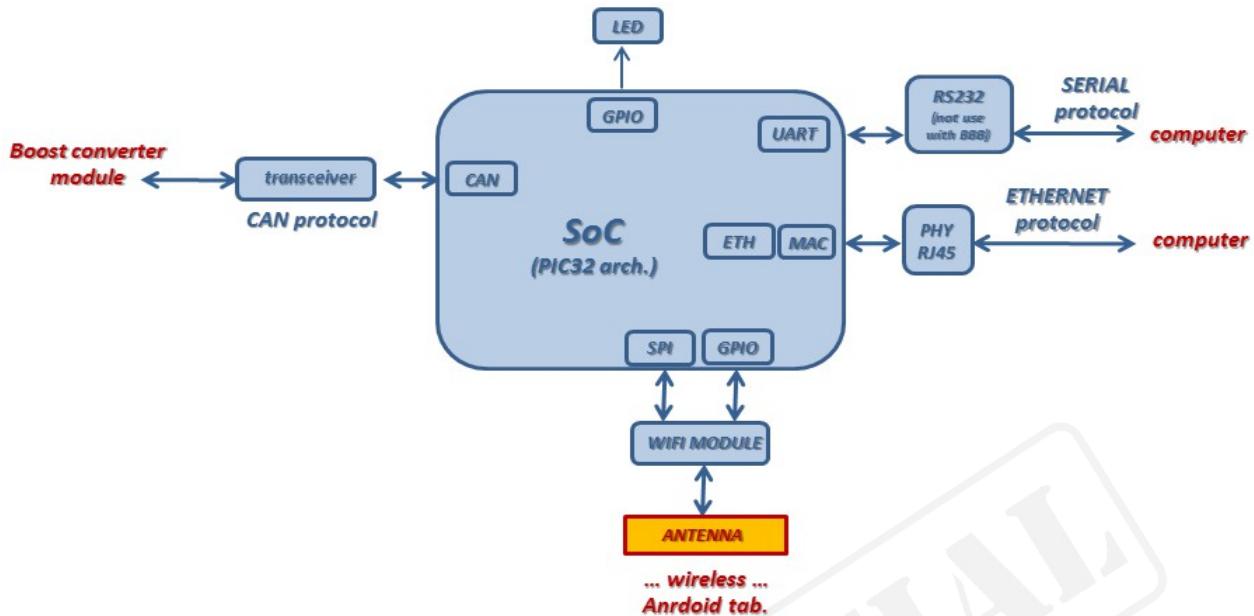
Solution très intéressante de notre côté construite sur une solution Linux Embarqué. La plateforme utilisé serait un SoC AM33xx de la famille Sitara proposé par Texas Instruments (sur plateforme BeagleBone Black ou BBB). Comme précisé précédemment, une distribution peut vous être actuellement fournie (sans le support côté kernel du CAN). Une distribution custom incluant drivers kernel CAN en cours de réalisation chez un de nos prestataires.

Dans tous les cas, le matériel vous sera prêté en début de projet : deux cartes beaglebone black, 2 cartes microSD 4Gb, 2 lecteur USB de cartes microSD, 2 câbles 3,3V série vers USB et deux modules d'interface CAN bus cape (cf. ci-dessous) :



5.2. Interfaces

Intéressons nous aux interfaces de communications de l'application. Observons l'architecture matérielle de l'application (pour MCU PIC32MX) :



- **Interface CAN** : l'interface CAN a été présentée plus en détail dans la partie BUS CAN de ce même document.
- **Interface LED** : L'application devra détecter la présence ou non d'un module boost (convertisseur + MCU) connecté. En l'absence de module la LED devra clignoter avec une périodicité de 500ms (500ms sec allumée, 500ms sec éteinte ...). En cas de présence la LED restera allumée.
- **Interface RESEAU IP-ETHERNET (RJ45)** : l'interface ethernet RJ45 proposera un serveur TCP (ou UDP ou les deux) embarqué côté SoC. Toutes les trames échangées se feront via échanges de chaînes de caractères (aucun format entier ou flottant). **Côté ordinateur, si vous n'avez pas le temps de proposer une interface graphique de communication (interface GTK, QT ...), votre application devra par défaut proposer la même interface que le module UART (cf. shell).** L'interface sera par exemple accessible depuis un client TCP côté ordinateur (par exemple Telnet). Pour la partie ethernet RJ45, nous travaillerons en IP fixe reconfigurable par liaison série.
- **Interface RESEAU IP-WIFI** : l'interface WIFI proposera un serveur TCP (ou UDP ou les deux) embarqué côté SoC. Toutes les trames échangées se feront via échanges de chaînes de caractères (aucun format entier ou flottant). Une présentation plus détaillée de la partie Android et interface utilisateur sera faite par la suite (partie INTERFACE UTILISATEUR). **Si possible, deux mode de fonctionnement doivent être supportés. Le mode AD HOC (imposé, doit fonctionner) et si possible le mode infrastructure ou Accès point (sans SSID).** Dans tous les cas, toutes connexions doit se faire vers le host sans identification. Le mode de fonctionnement (ad hoc ou infrastructure) sera géré à la compilation et ne sera pas modifiable en cours de fonctionnement.

- **Interface UART :** L'application devra proposer un shell afin de communiquer voir reconfigurer sans recompilation des variables d'environnement du firmware embarqué. L'application devra proposer une interface de communication (cf. shell) implémentant un jeu de commande prédéfini dans le firmware. Jeu de commande à supporter :
 - **eth** : passe en mode ethernet et stop le mode wifi. Le serveur TCP (ou UDP ou les deux) embarqué côté SoC duplique l'interface UART vers l'ordinateur de supervision (client TCP ou UDP ou les deux).
 - **wifi** : passe en mode wifi et stop le mode eth. La seule communication réseau IP active est la communication WIFI vers la tablette
 - **ip** : modification de l'adresse IP pour le mode ethernet. 4 valeurs entière séparées par un point , par exemple : 192.168.94.1
 - **netmask** : modification du masque de sous réseau pour le mode ethernet. 4 valeurs entière séparées par un point , par exemple : 255.255.255.0
 - **config** : affiche les principales informations IP du mode courant (ethernet ou wifi). Par exemple : adresse IP, masque de sous réseau, ports TCP et UDP, mode de fonctionnement (ad hoc ou infrastructure) ... ne pas hésiter à rajouter des champs si ils vous semblent importants.
 - **list** : affiche tous les modules boost connectés ainsi les principales informations techniques concernant chaque module. Chaque module apparaîtra sous le nom suivant boost0, boost1 ... (le premier connecté étant le 0, puis le 1 ... etc). Nous afficheront également les valeurs moyennes de la tension de sortie, la tension d'entrée, le courant d'entrée, la puissance en entrée et le courant estimé de sortie (rendement unitaire). Affiche "none" si aucun module n'est connecté.
 - **help** : retourne le jeu de commandes supportées ainsi qu'une courte aide

L'interface proposée par le SoC devra absolument respecter l'affichage suivant (exemple de d'implémentation du jeu de commandes) :

cf. page suivante

```
***** * * ***** * **** *   ***   *****   **   ***** ***   ***
*   **** * * *** *** * *** *   *   *   ***   *   *   *   *** *   *   *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
***** * *   *   *   *   *   ***   ***** * *   *   *   ***** * *   *   *
```

*citroen corp. with your company partnership
 boost converter supervision module interface. Enter help to know the
 supported commands*

```
boost:~# help
eth      start ethernet mode and stop wifi mode
wifi    start wifi mode and stop ethernet mode
ip       update ip value for ethernet module
netmask  update netmask value for ethernet module
config   print ip informations for running mode
list     print informations about connected boost modules
help     print supported commands
boost:~# eth
ethernet mode is running
boost:~# ip
update ip value for ethernet module (??.?.?.?) :
172.24.123.13
ip value is updated
boost:~# ip
update ip value for ethernet module (??.?.?.?) :
172.24.123.13azerty
error: failed value
boost:~# netmask
update netmask value for ethernet module (??.?.?.?) :
255.255.0.0
netmask value is updated
boost:~#
```

```
boost:~#
სეტვარქ ჩაწინ ჯე იხდათხა
522°522°0°0
იხდათხ სეტვარქ ჩაწინ ჯო ცემენტ შეინი
```

```
boost:~# netmask
update netmask value for ethernet module (??.?.?.?) :
255.255.0.0azerty
error: failed value
boost:~# config
ethernet module ip configuration :
ip      172.24.123.13
netmask  255.255.0.0
you can print other importants config !
tcp port  5151
udp port  1414
boost:~# wifi
wifi mode is running
boost:~# config
ethernet module ip configuration :
mode    ad hoc
ip      192.168.92.1
netmask  255.255.255.0
you can print other importants config !
tcp port  5151
udp port  1414
boost:~# list
boost0    Vi=12.1V Il=1.5A Pi=18.15W Vo=24.0V Io=750mA (rendement#1)
Boost1    Vi=11.4V Il=1.2A Pi=13.68W Vo=24.0V Io=570mA (rendement#1)
boost:~# azerty
error: enter help to know the supported commands
boost:~#
```

```
boost:~#
სულ: გერბ ფი გამო ფუ გუბისტები კომანდები
boost:~# ძალები
boost:~#
```

Cahier des charges :

- *Les échéances seront fixées en cours de projet*
- *Proposer une modélisation de l'architecture logicielle de l'application (environnement multi-tâches, outils de synchronisation, communication, protection). La modélisation pourra être fournie avec une convention graphique propre à votre entreprise ou sinon devra respecter les conventions du standard SysML (Systems Modeling Languages). A envoyer en cours de projet.*
- *Votre solution logicielle devra être disponible et téléchargeable via internet à chaque moment du projet (mise en place dès le 3 jours de projet). Nous vous demandons d'utiliser des solutions (temporairement) ouverte via dépôt distant sur GitHub (interface familière du côté de nos équipes). Il faudra donc un dépôt GitHub unique pour le firmware propre à cette partie de l'application.*

CONFIDENTIAL

6. INTERFACE UTILISATEUR

6.1. Système de contrôler/supervision

L'une des parties les plus importantes du projet est celle qui suit, car révélatrice du bon fonctionnement de toute la chaîne de traitement de l'information précédemment implémentée. L'interface utilisateur doit garantir une interface simple de contrôle/supervision du convertisseur boost. Cette interface se fera soit depuis un ordinateur soit depuis une tablette (ce démonstrateur doit être fonctionnel), soit les deux.

6.2. Interface ordinateur

- **solution n°1 :**

si vous n'avez pas le temps de développer une interface graphique évoluée côté ordinateur, l'application devra au pire réaliser une redirection de l'interface UART (serveur TCP ou UDP ou les deux) vers un client côté ordinateur. Une validation simple pourra se faire avec Telnet pour le client TCP.

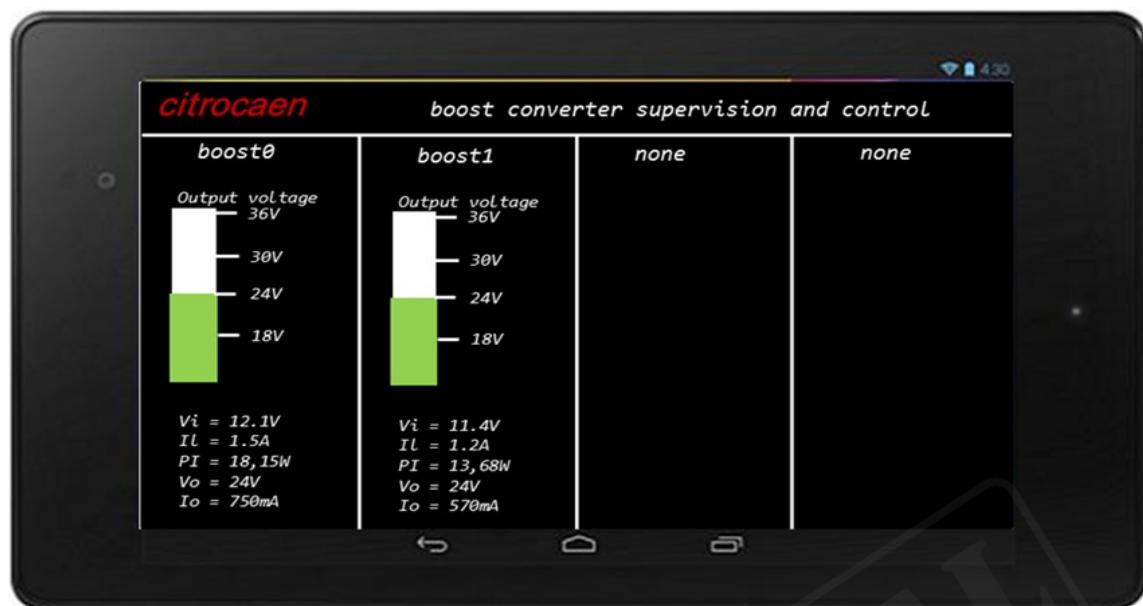
- **solution n°2 :**

Si le temps le permet, proposer un IHM pour l'utilisateur, développée par exemple sous GTK (sous Liux) ou QT ... ou autre

6.3. Interface Android

Nous vous demandons de développer, valider et porter sur l'Android market une application Android (non évolutive dans un premier, évolutive si possible). Cette application simple et conviviale proposera à l'utilisateur un IHM lui permettant de superviser voir reconfigurer à distance la tension de sortie du convertisseur (valeurs admissibles 18V ou 24V ou 30V ou 36V). Voici ci-dessous un exemple d'IHM, bien entendu si votre graphiste peu proposer une interface encore plus conviviale et évolutive (limitation actuelle à 4 modules), ce ne peut être qu'un plus.

cf. page suivante



Io = 520mA
 Vo = 24V
 Li = 18.12H
 Ti = 1.5ms

L'affichage devra être réactualisé toutes les 500ms. Le slider permettant l'affichage de la tension de sortie du boost sera actif en acquisition et permettra de modifier via l'interface tactile la valeur de la tension sortie du convertisseur boost ciblé.

Cahier des charges :

- **Les échéances seront fixées en cours de projet**
- **Développer l'interface proposé ci-dessus ou une interface le plus proche possible de celle-ci.**
- **Toute proposition d'interface plus conviviale et plus évoluée sera le bien venu. Ne pas hésiter à soumettre votre projet.**

7. DOCUMENTATIONS TECHNIQUES

7.1. Schémas carte de puissance

7.2. Schémas carte PIC32 MAXI WEB

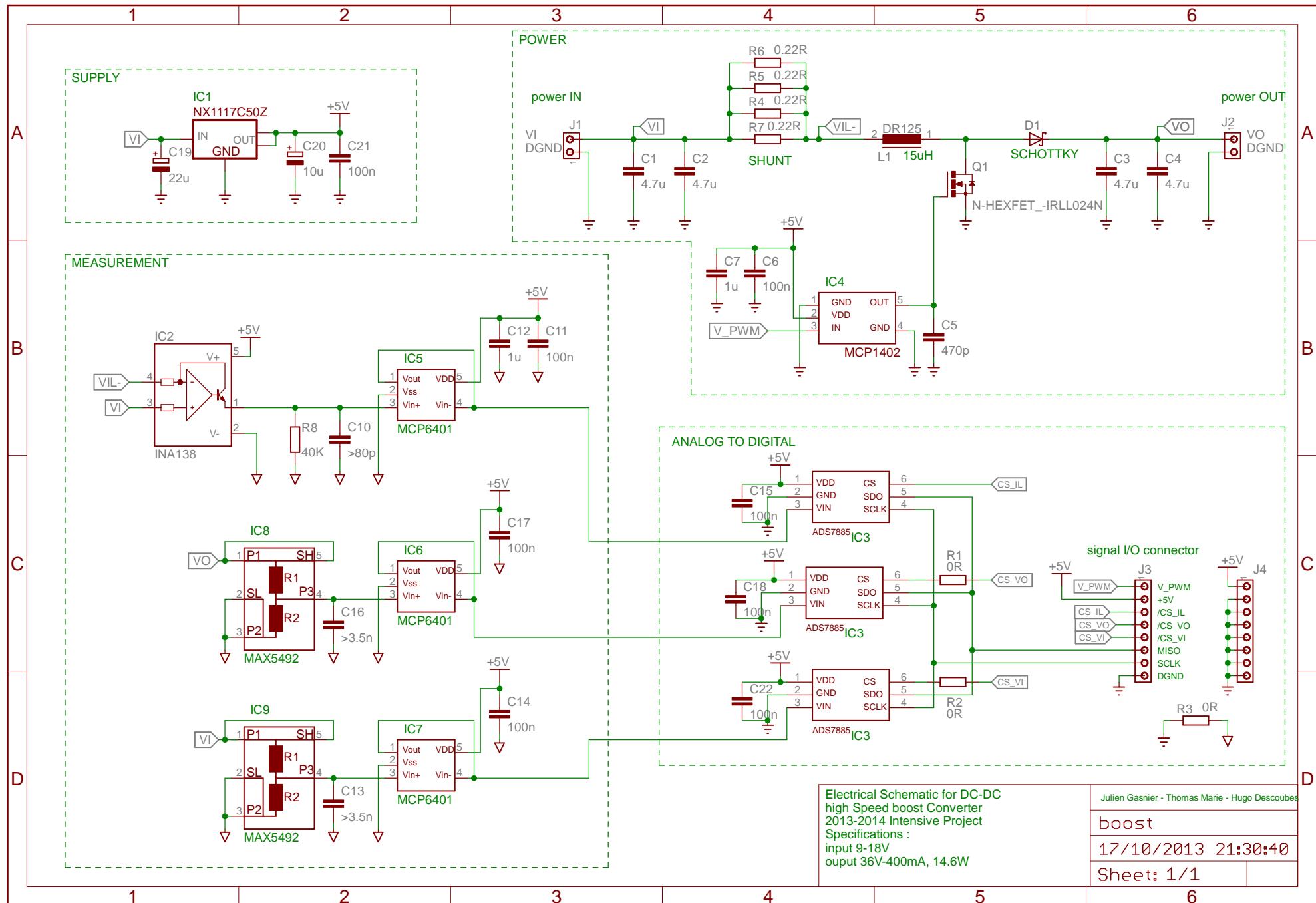
7.3. Schémas carte BeagleBone Black

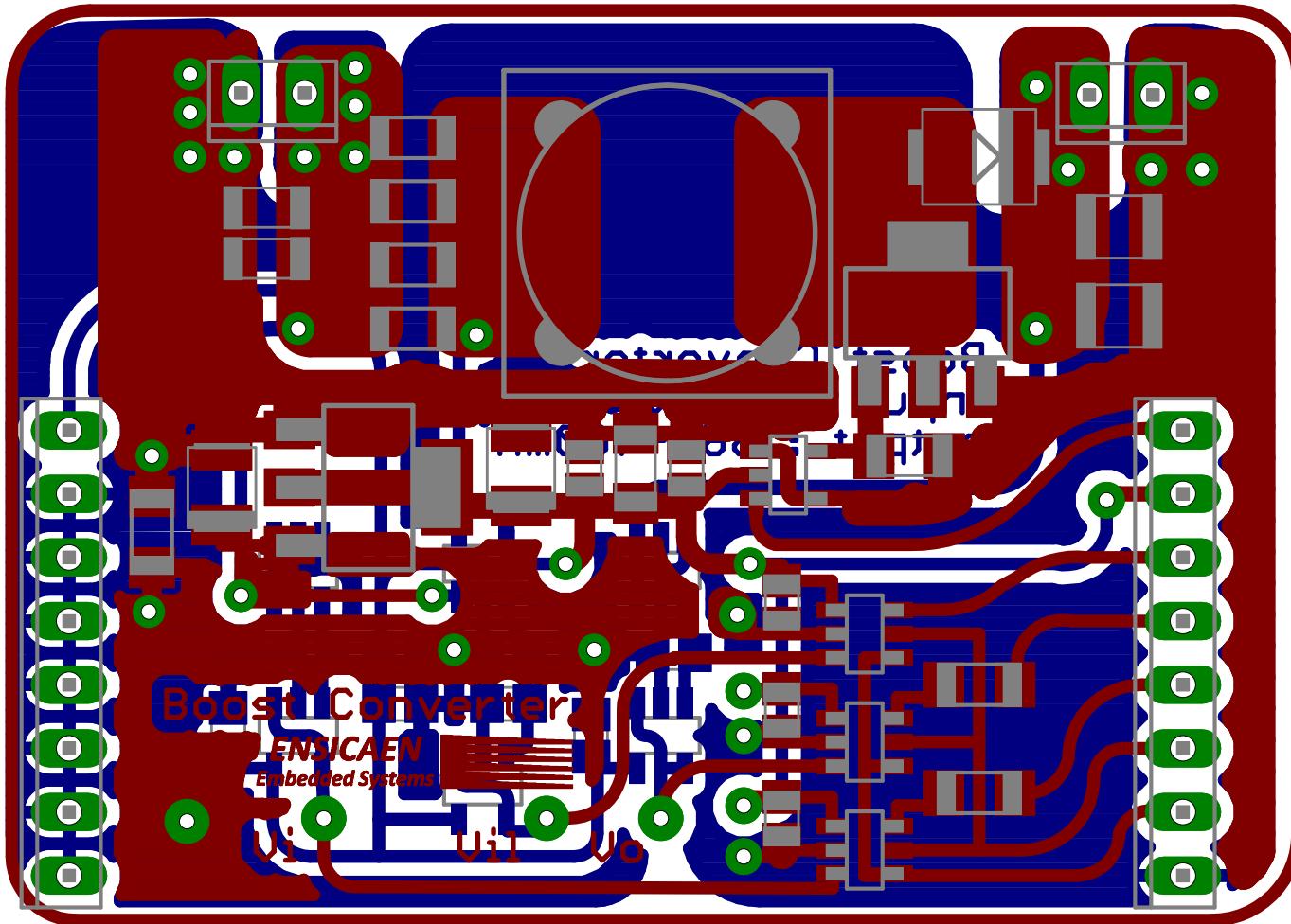
7.4. liens utiles

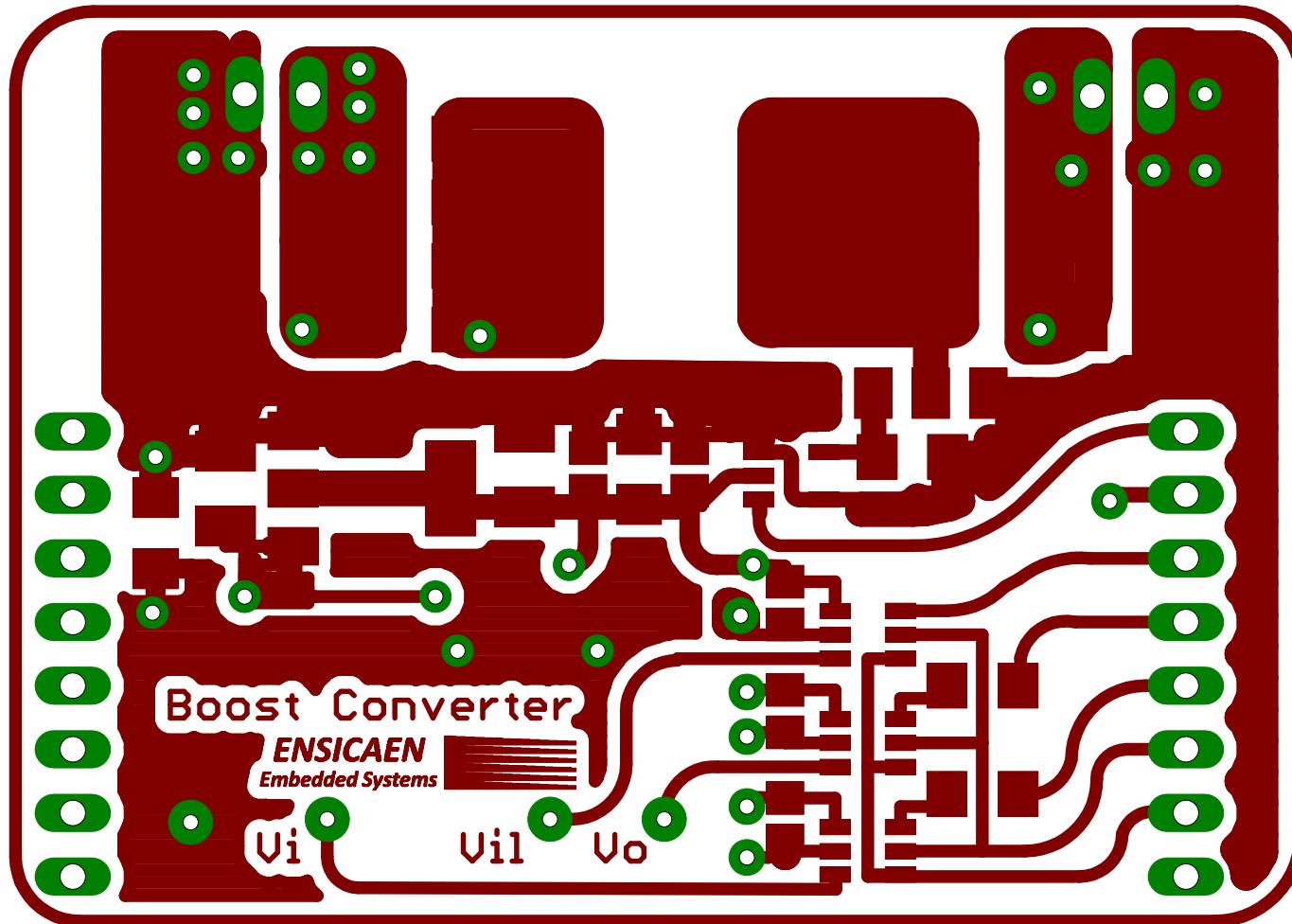
CONFIDENTIAL

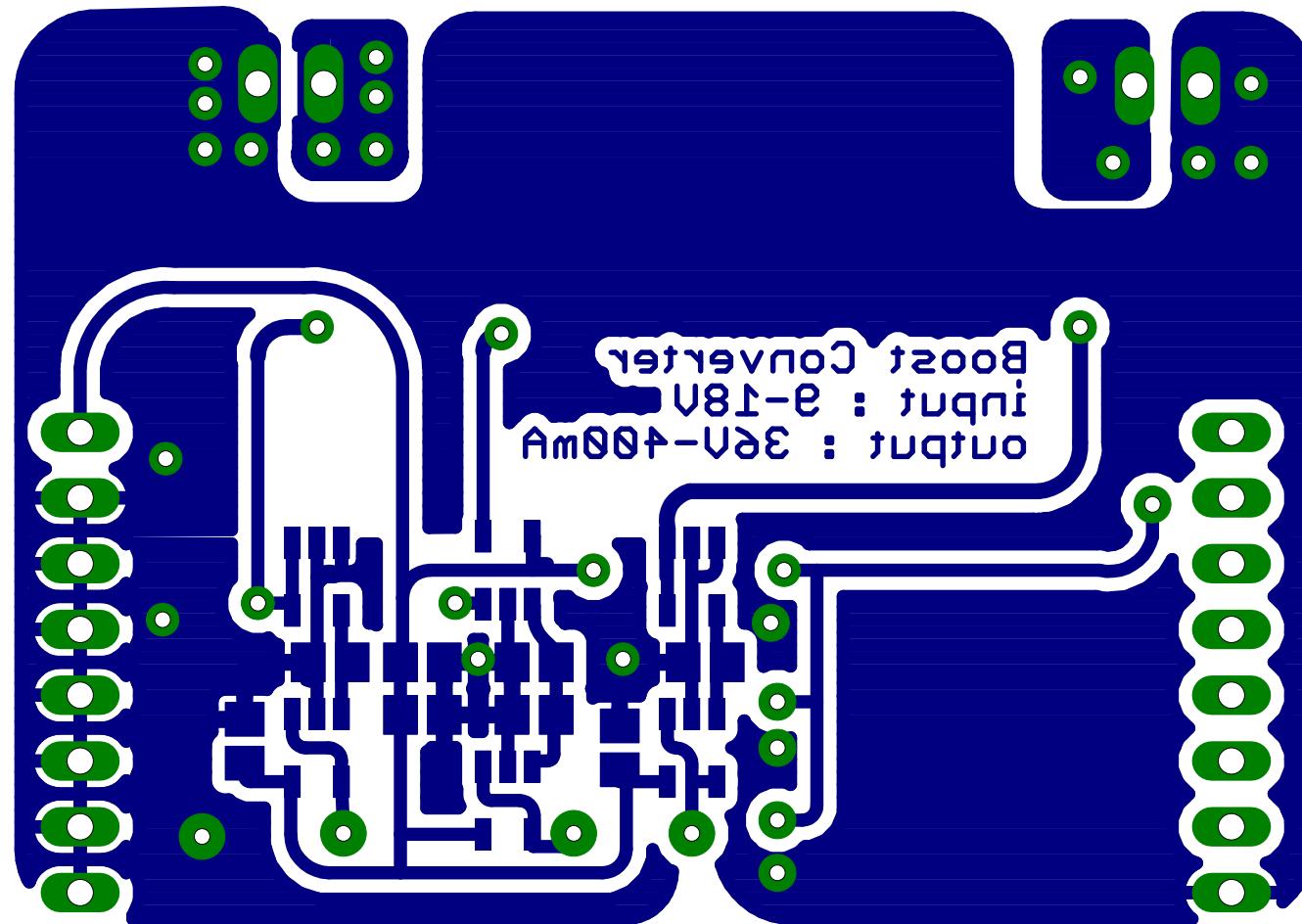
7.1. SCHEMAS CARTE PUISSANCE

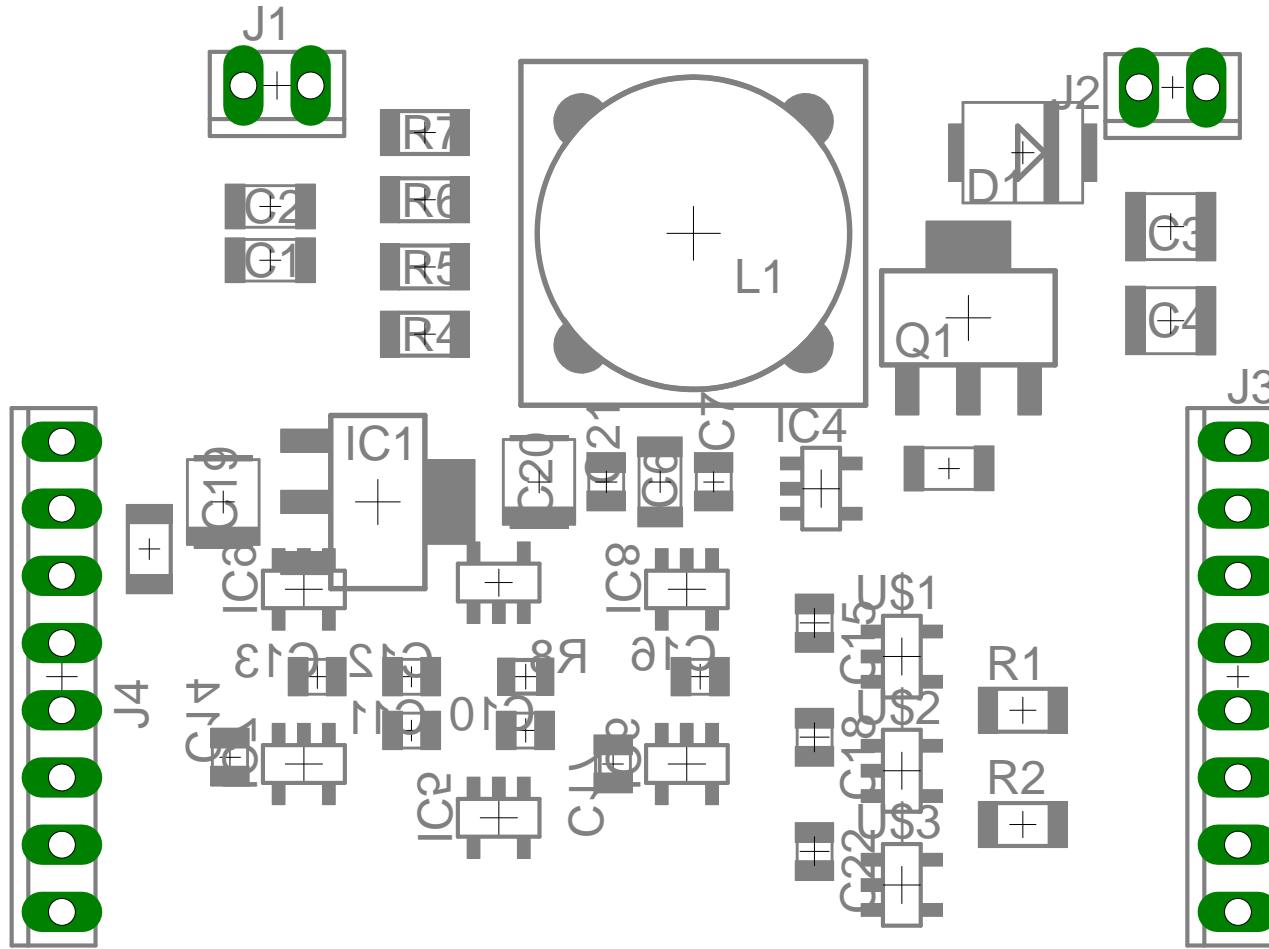
CONFIDENTIAL





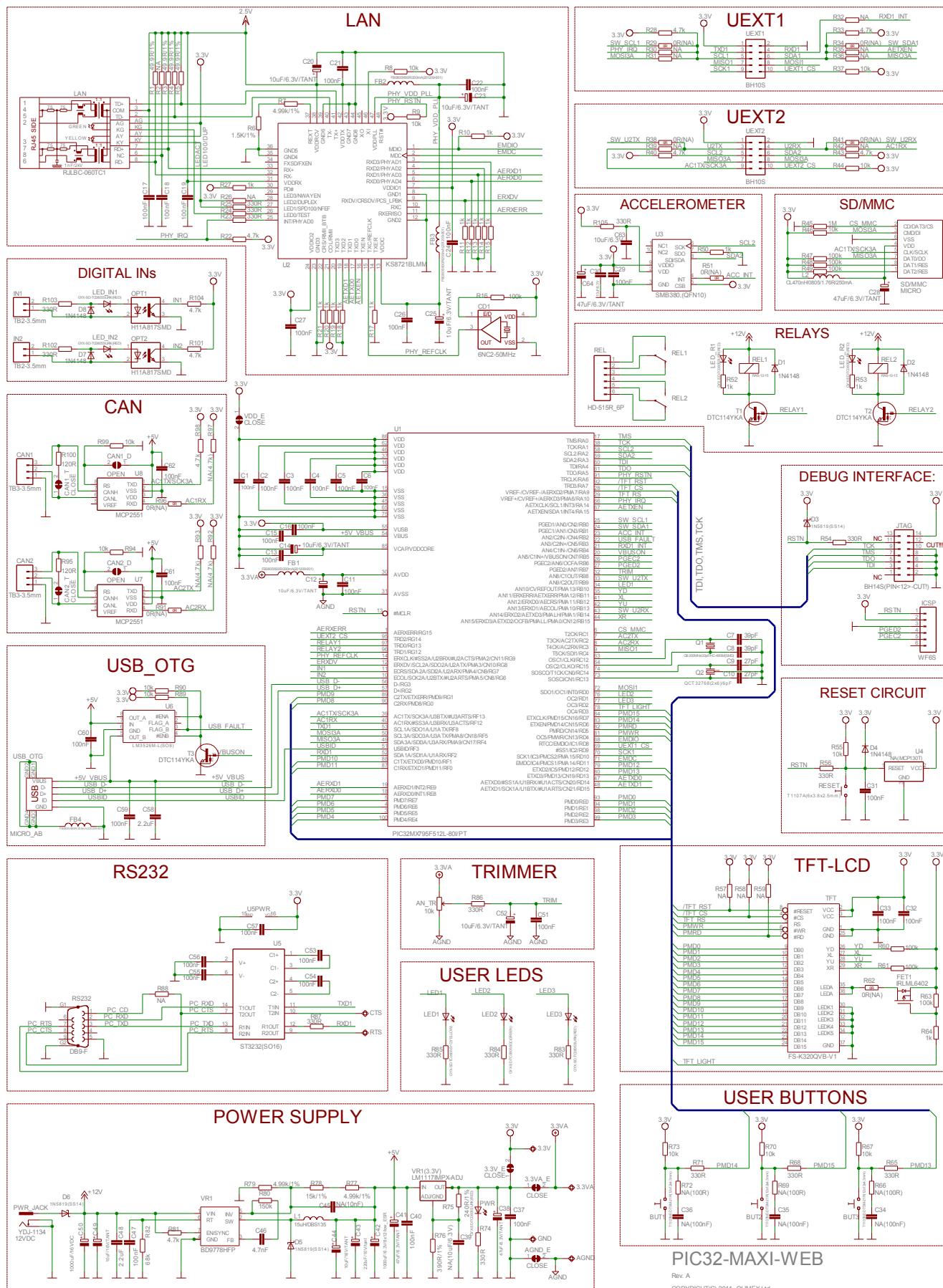






7.2. SCHEMAS CARTE PIC32 MAXI WEB

CONFIDENTIAL



PIC32-MAXI-WEB

Rev. A

COPYRIGHT(C) 2011, OLIMEX Ltd.

<http://www.olimex.com/dev>

7.3. SCHEMAS CARTE BEAGLEBONE BLACK

CONFIDENTIAL

REV	Description	DATE	BY	PAGE NO.	SCHEMATIC PAGE
A4A	Initial production Release.	11/19/2012	GC	1	COVER PAGE
A5	On the initial production release the processors were to be found incorrect as supplied by TI. Parts while marked AM359 were actually AM352. This revision uses the correct parts.	1/2/2013	GC	2	POWER MANAGEMENT
A5A	1. Deleted R29-R44 from the LCD lines. 2. Added 47pf capacitors C156-C173 to LCD data lines to ground. 3. Changed schematic revision to A5A. 4. Changed a few footprints after PCB update for above changes. 5. Added access point for the battery function of the TPS65217C. 6. Added Ferrite beads in series with LED power and 5V power rail of the USB host connector. Required to pass FCC/CE testing due to noise emissions on that pin. 7. Added power button to enable sleep, wakeup, power down and power up features on the system. 8. Added Modification to add 100K ohm resistor to ground to prevent crosstalk when serial cable is not plugged in.	2/8/2013	GC	3	PROCESSOR 1 OF 3, JTAG HEADER
A5B	1. Added 100K pulldown on J1 pin 4 to prevent crosstalk when serial cable is not connected into PCB layout. 2. Changed the LED resistors to 4.75K to lower the brightness.	5/21/2013	GC	4	PROCESSOR 2 OF 3, UAB PORTS
A5C	1. Changed R46, R47, R48 to 0 ohms. 2. Changed R45 to 22 Ohms. Change was made due to production failures on some boards due to differences in impedances.	6/12/2013	GC	5	PROCESSOR 3 OF 3
				6	LED, CONFIGURATION AND BUTTON
				7	DDR3 MEMORY
				8	eMMC FLASH
				9	10/100 ETHERNET
				10	HDMI FRAMER
				11	EXP CONN, uSD

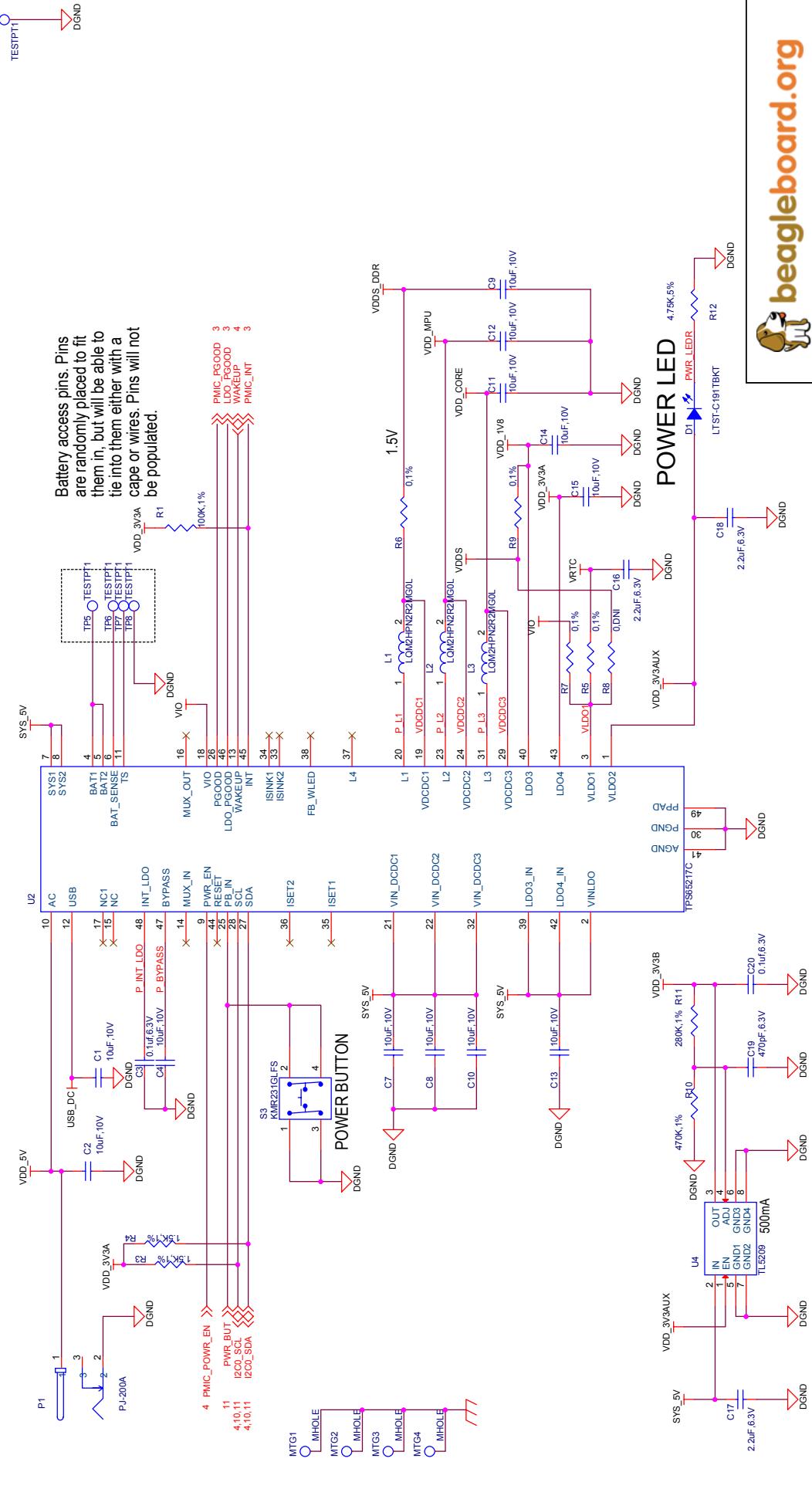
NOTE: PCB Revision for this board is Rev B4.

This schematic is *NOT SUPPORTED* and DOES NOT constitute a reference design. Only "community" support is allowed via resources at BeagleBoard.org/discuss.

THERE IS NO WARRANTY FOR THIS DESIGN, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE DESIGN "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE DESIGN IS WITH YOU. SHOULD THE DESIGN PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

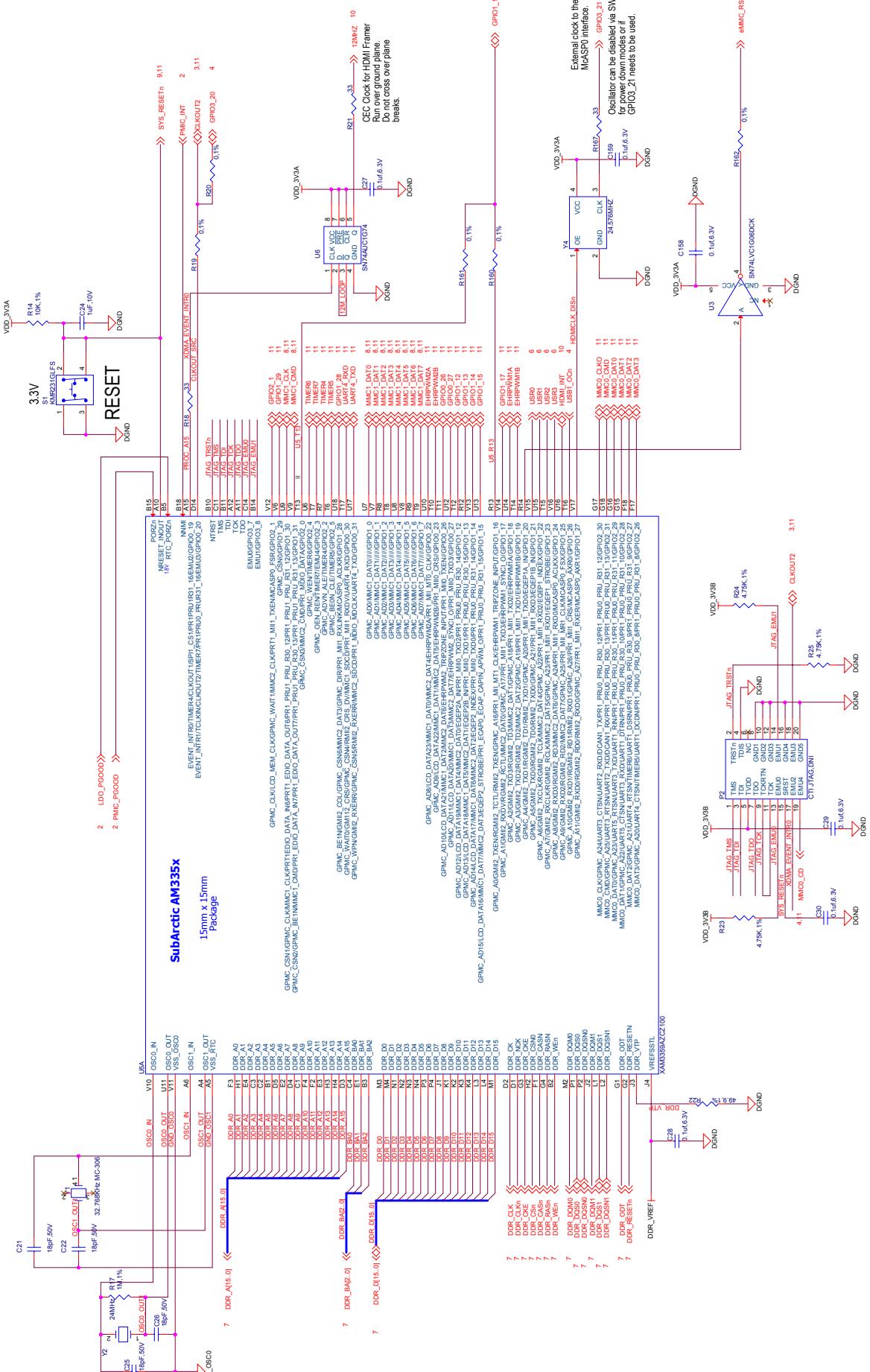


Title	BeagleBone Black Cover Page	Rev
B	450-5500-001	A5C
Date: Wednesday, June 12, 2013	Sheet 1 of 11	

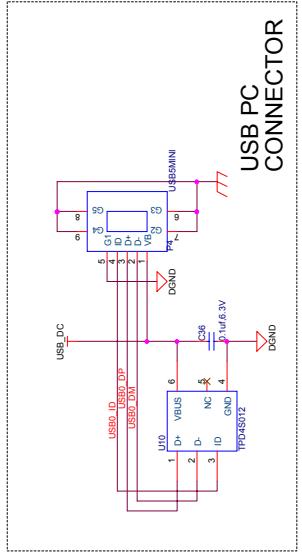
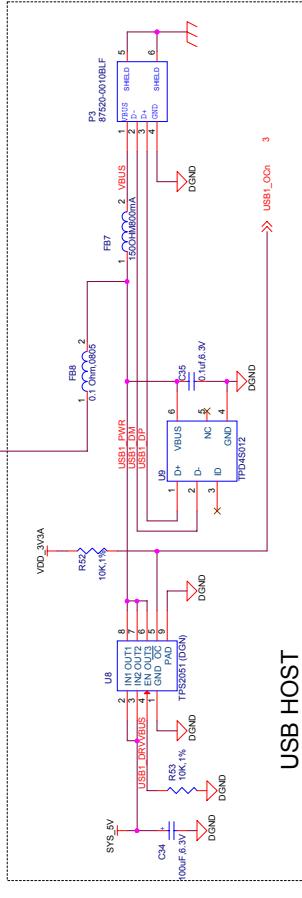
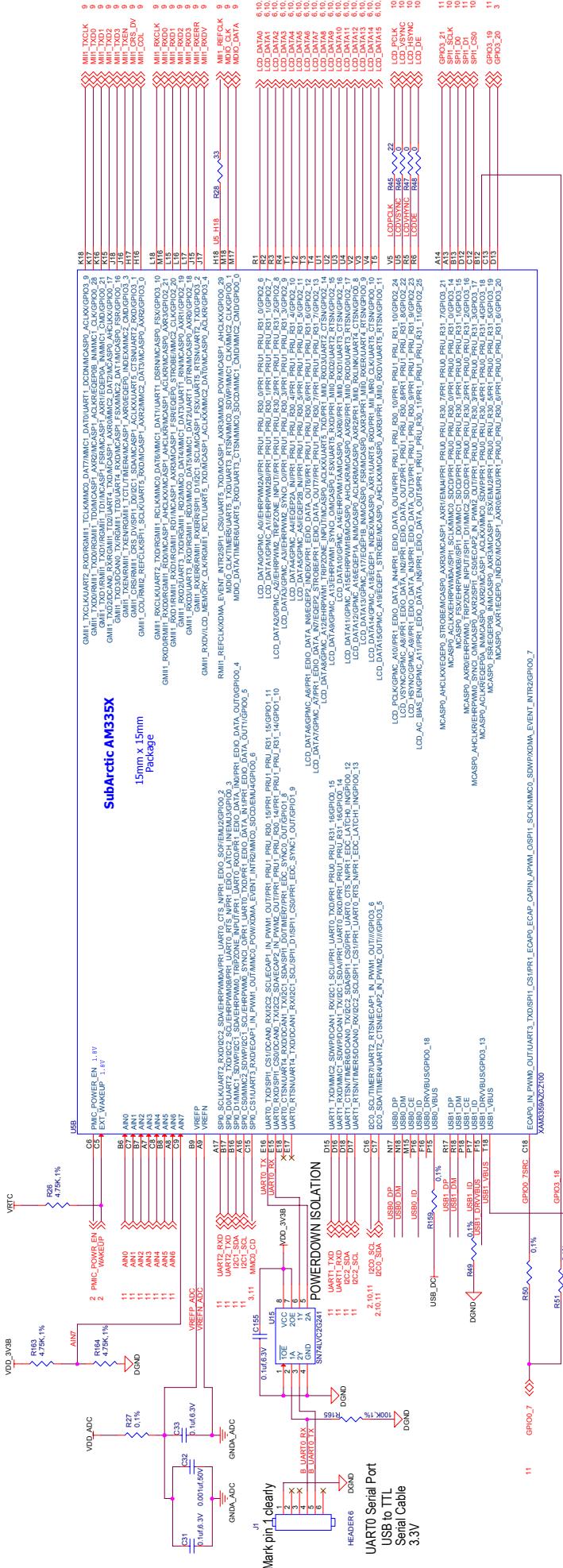


beagleboard.org

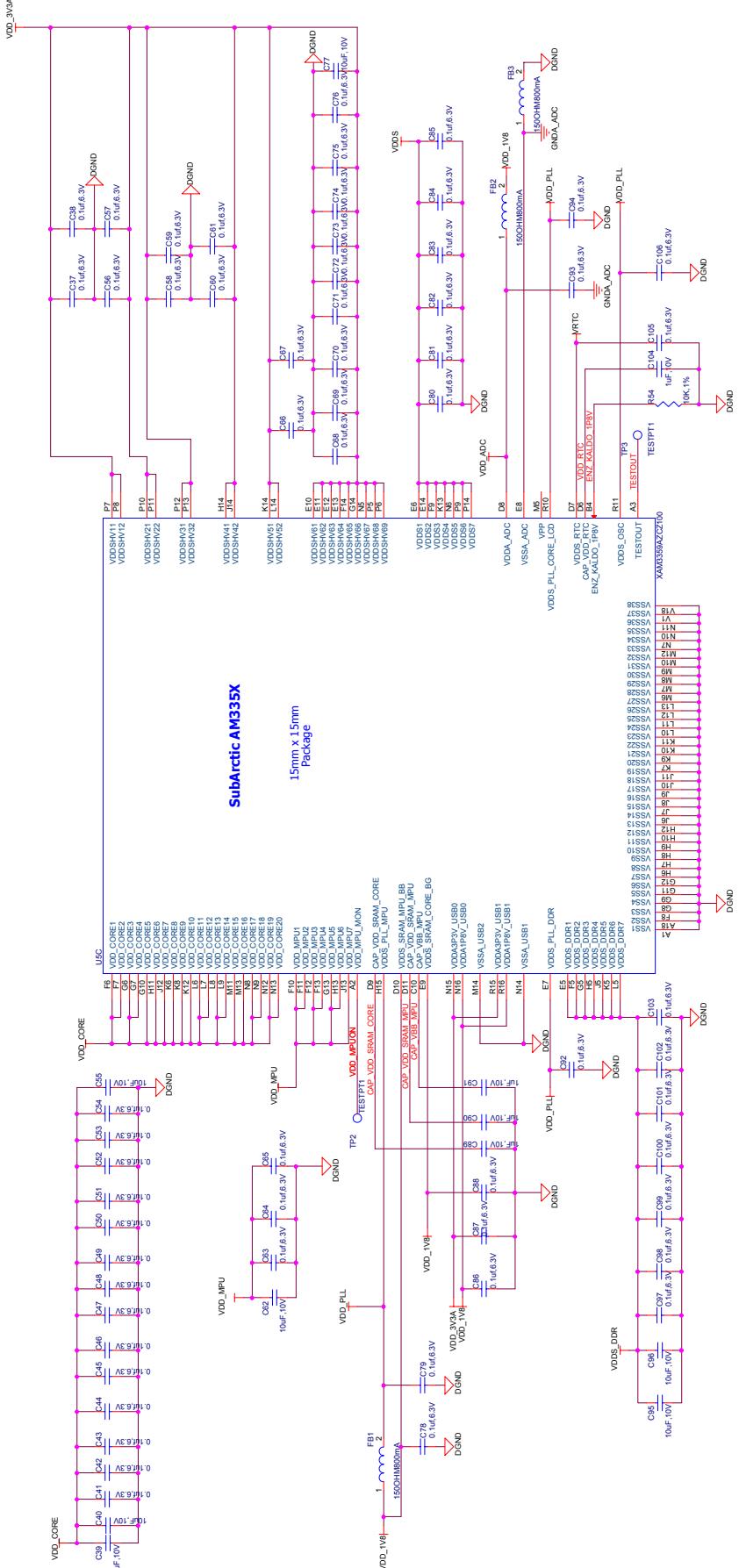
Title	BeagleBone Black Power Management		
Size	Document Number		
B	450-5500-001		
Date:	Wednesday, June 12, 2013		
	Sheet	2	of 11
	Rev	A5C	

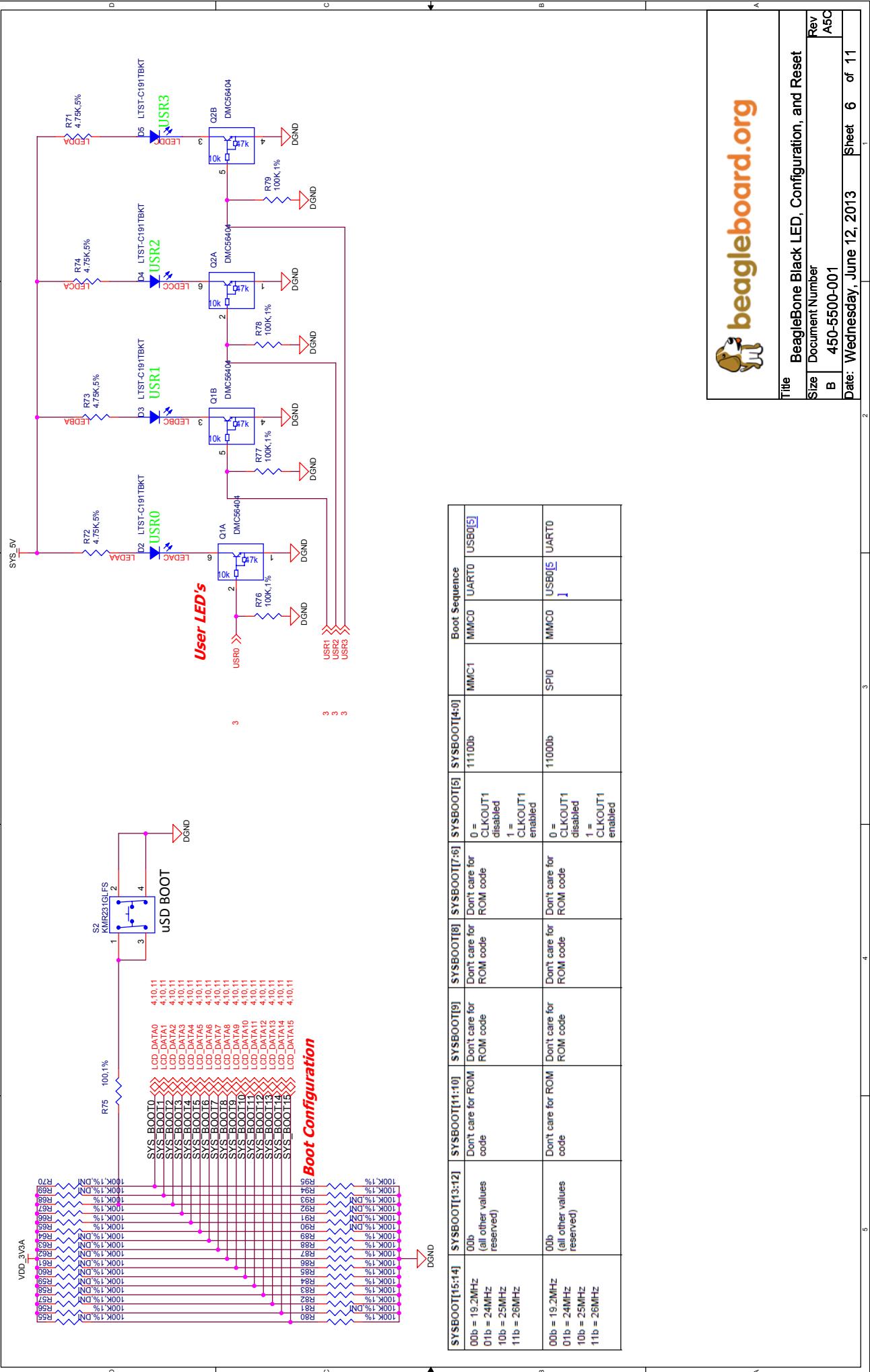


Title	BeagleBone Black Processor 1 of 3 and JTAG		
Size	Document Number:		
C	450-5500-001		
Date:	Wednesday, June 12, 2013		
	Sheet	3	of 11
	Rev	A5C	

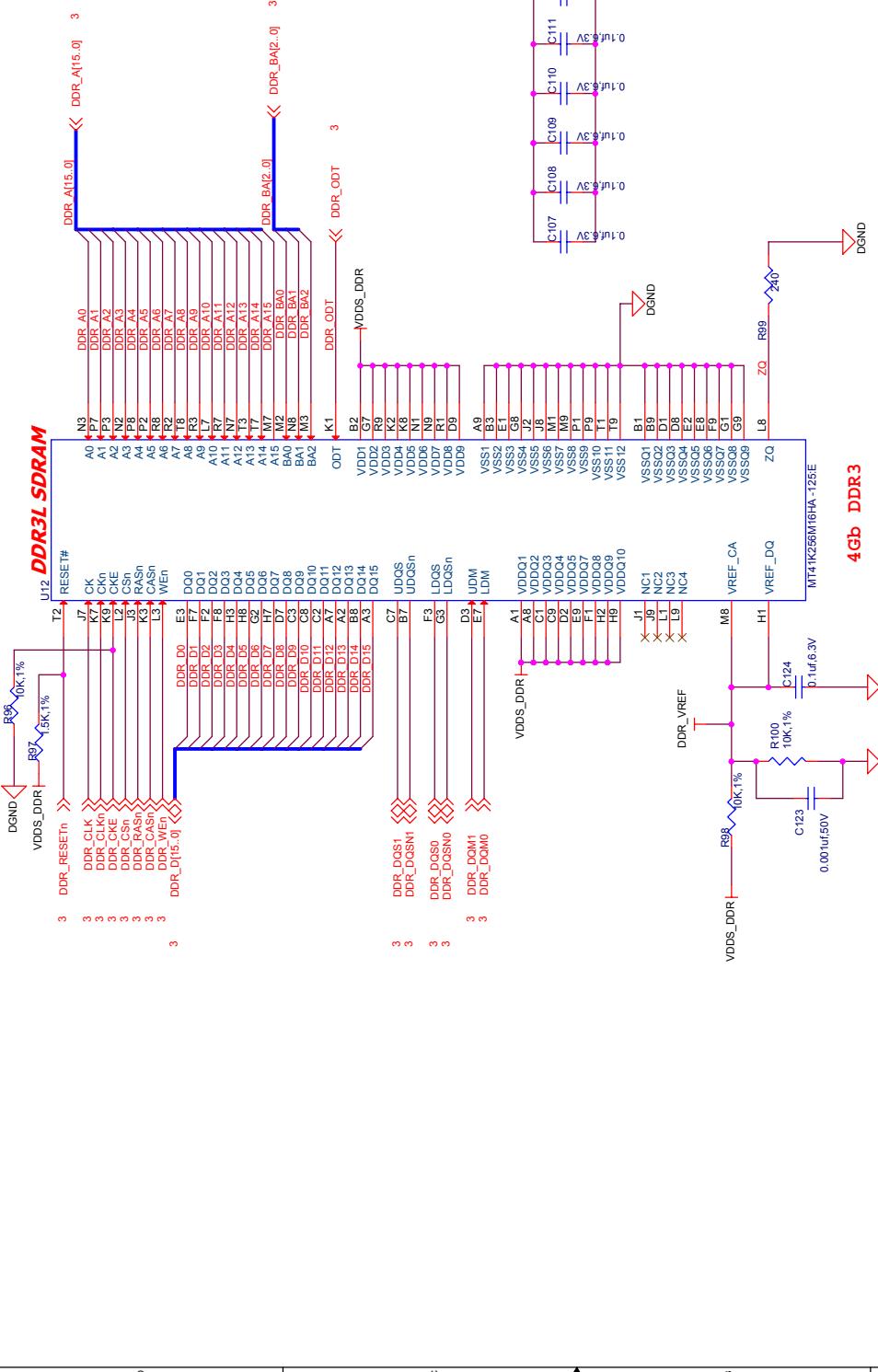


Title	BeagleBone Black Processor 2 of 3, USB, and Serial		
Size	Document Number	Rev	ASC
C	450-5500-001		
Date:	Wednesday, June 12, 2013	Sheet 4	of 11



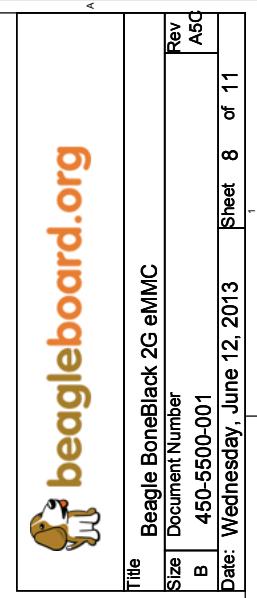
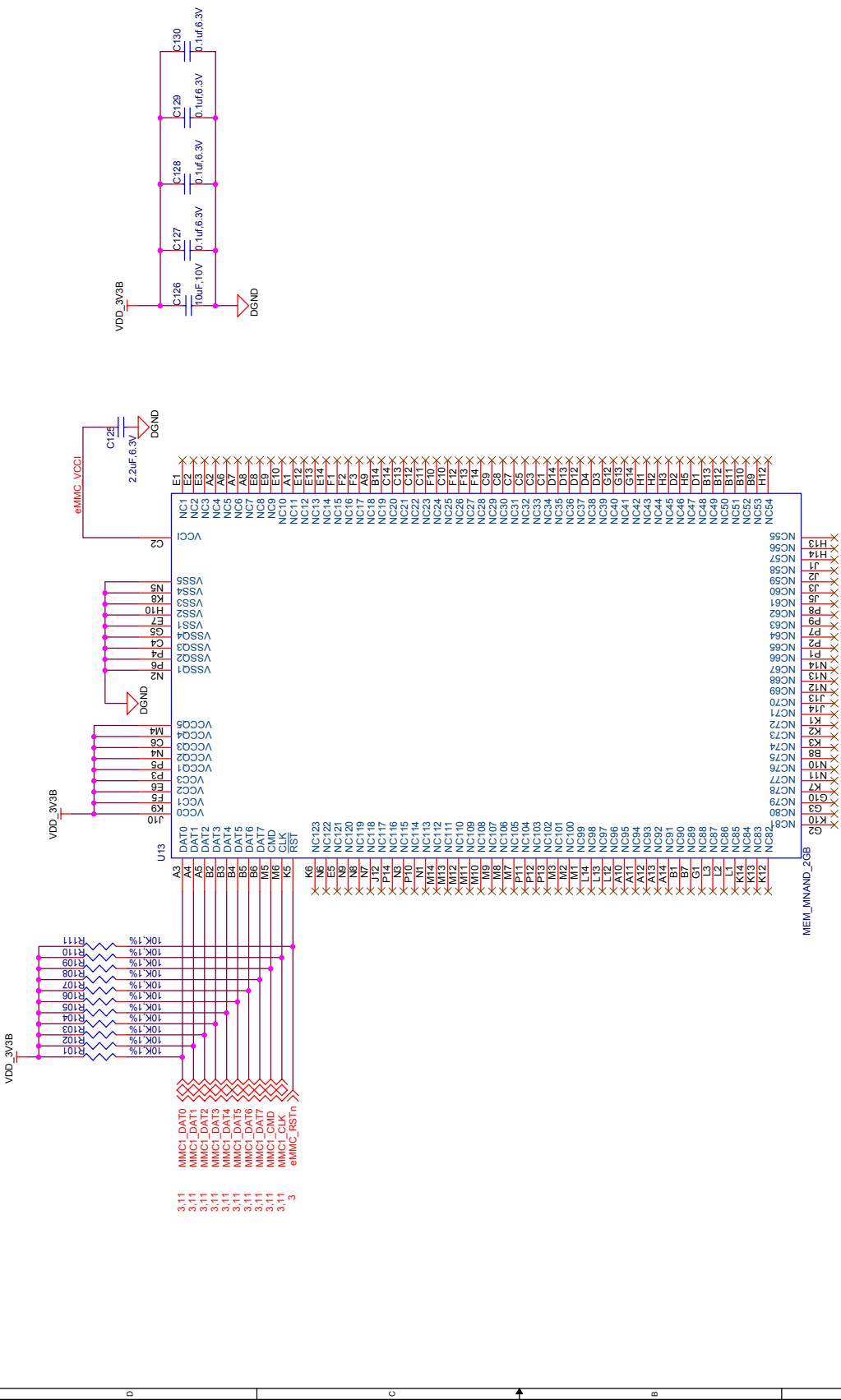


Title BeagleBone Black LED, Configuration, and Reset
Size Document Number Rev
B 450-5500-001 A5C
Date: Wednesday, June 12, 2013 Sheet 6 of 11



beagleboard.org

Title	BeagleBone Black DDR3 Memory		
Size	Document Number		
B	450-5500-001		
Date:	Wednesday, June 12, 2013		
	Sheet	7	of 11
	Rev	A5C	



Title Beagle BoneBlack 2G eMMC



beagleboard.org

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

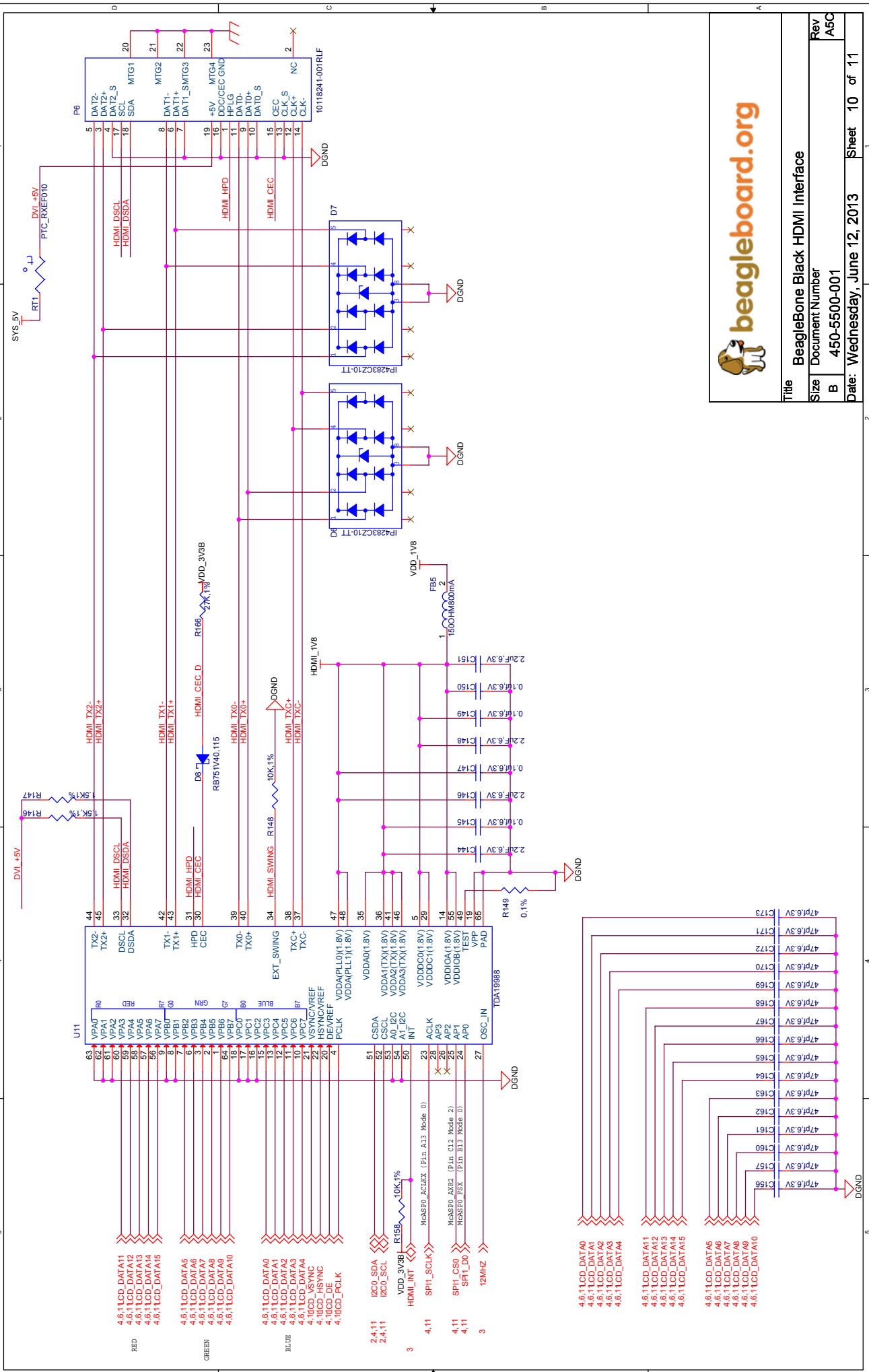
293

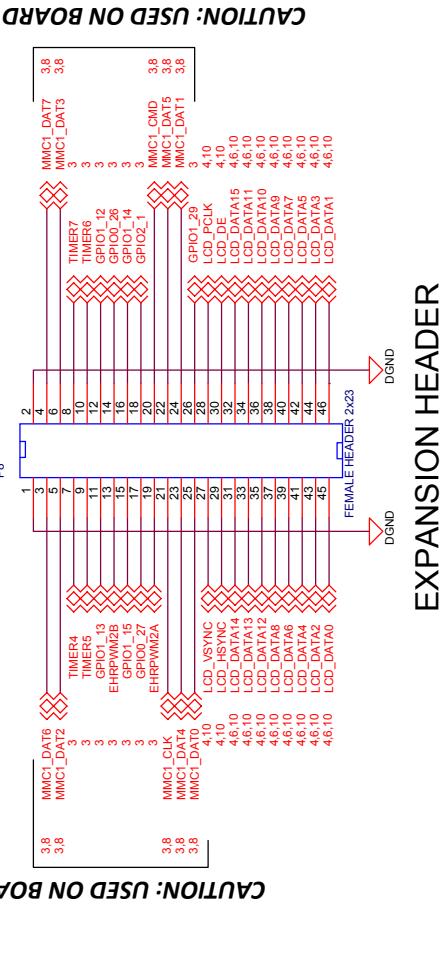
294

295

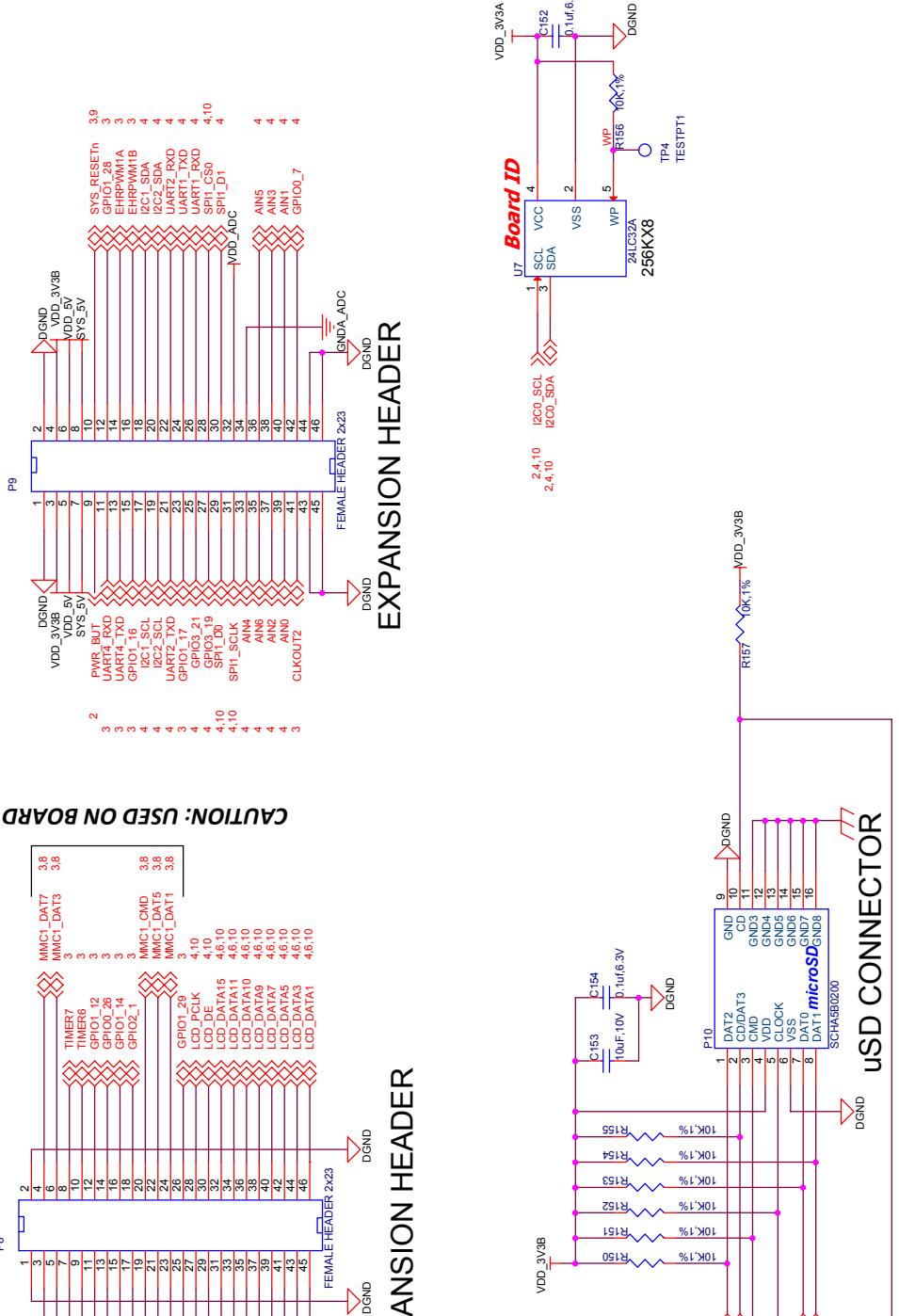
296

</





EXPANSION HEADER



beagleboard.org

The BeagleBone Black Expansion Headers, uSD and EEPROM
 Size Document Number Rev
 B 450-55-00-001 A5C
 Date: Wednesday, June 12, 2013 Sheet 11 of 11

7.4. Liens utiles

- **projet pic32 maxi web :**

<https://www.olimex.com/Products/PIC/Development/PIC32-MAXI-WEB/>

<https://www.olimex.com/Products/Modules/Ethernet/MOD-WIFI/>

- **projet beaglebone :**

<http://beagleboard.org/Products/BeagleBone+Black/>

<http://circuitco.com/support/index.php?title=BeagleBoneBlack>

http://circuitco.com/support/index.php?title=BeagleBone_Black_FAQ

http://circuitco.com/support/index.php?title=BeagleBone_Capes

http://circuitco.com/support/index.php?title=BeagleBone_CANBus

- **projet beaglebone (building kernel) :**

http://wiki.beyondlogic.org/index.php/BeagleBoneBlack_Building_Kernel

<http://eewiki.net/display/linuxonarm/BeagleBone>

kernel linux 3.8.13 patché (pour projet beaglebone black) avec installation driver CAN (buildin) :
<https://github.com/embeddedEnsicaen/kernel>

- **android :**

<http://developer.android.com/index.html>

- **protocole CAN (version 2.0) :**

<http://esd.cs.ucr.edu/webres/can20.pdf>

http://www.bosch-semiconductors.de/en/ubk_semiconductors/safe/ip_modules/m_can/m_can.html